PERCONA

www.percona.com

Percona XtraBackup 2.2 Documentation

Release 2.2.3

Percona LLC and/or its affiliates

June 12, 2014

CONTENTS

1 Introduction **3**
1.1 About Percona Xtrabackup . . . 3

2 Installation **7**
2.1 Installing *Percona XtraBackup* from Binaries . . . 7
2.2 Compiling and Installing from Source Code . . . 10

3 User's Manual **13**
3.1 *Percona XtraBackup* User Manual . . . 13

4 Tutorials, Recipes, How-tos **63**
4.1 How-tos and Recipes . . . 63

5 Miscellaneous **83**
5.1 Frequently Asked Questions . . . 83
5.2 *Percona XtraBackup* Release Notes . . . 84
5.3 Glossary . . . 115
5.4 Index of files created by Percona XtraBackup . . . 116
5.5 Trademark Policy . . . 118

6 Indices and tables **119**

Index **121**

Percona XtraBackup is an open-source hot backup utility for *MySQL* - based servers that doesn't lock your database during the backup.

It can back up data from *InnoDB*, *XtraDB*, and *MyISAM* tables on *MySQL* 5.1 [1], 5.5 and 5.6 servers, as well as *Percona Server* with *XtraDB*. For a high-level overview of many of its advanced features, including a feature comparison, please see *About Percona Xtrabackup*.

Whether it is a 24x7 highly loaded server or a low-transaction-volume environment, *Percona XtraBackup* is designed to make backups a seamless procedure without disrupting the performance of the server in a production environment. Commercial support contracts are available.

Percona XtraBackup is a combination of the **xtrabackup** *C* program, and the **innobackupex** *Perl* script. The **xtrabackup** program copies and manipulates *InnoDB* and *XtraDB* data files, and the *Perl* script enables enhanced functionality, such as interacting with a running *MySQL* server and backing up *MyISAM* tables.

Note: This is a **BETA** quality release and it is not intended for production.

---

[1] Support for InnoDB 5.1 builtin has been removed in *Percona XtraBackup* 2.1

CHAPTER

ONE

INTRODUCTION

1.1 About Percona Xtrabackup

Percona XtraBackup is the world's only open-source, free *MySQL* hot backup software that performs non-blocking backups for *InnoDB* and *XtraDB* databases. With *Percona XtraBackup*, you can achieve the following benefits:

- Backups that complete quickly and reliably
- Uninterrupted transaction processing during backups
- Savings on disk space and network bandwidth
- Automatic backup verification
- Higher uptime due to faster restore time

Percona XtraBackup makes *MySQL* hot backups for all versions of *Percona Server*, *MySQL*, and *MariaDB*. It performs streaming, compressed, and incremental *MySQL* backups.

Percona XtraBackup works with *MySQL*, *MariaDB*, and *Percona Server*. It supports completely non-blocking backups of *InnoDB*, *XtraDB*, and *HailDB* storage engines. In addition, it can back up the following storage engines by briefly pausing writes at the end of the backup: *MyISAM*, *Merge*, and *Archive*, including partitioned tables, triggers, and database options.

Percona's enterprise-grade commercial MySQL Support contracts include support for *Percona XtraBackup*. We recommend support for critical production deployments.

1.1.1 MySQL Backup Tool Feature Comparison

Features	Percona XtraBackup	MySQL Enterprise Backup (InnoDB Hot Bac
License	GPL	Proprietary
Price	Free	$5000 per server
Open source	Yes	
Streaming and encryption formats	Open source	Proprietary
Supported MySQL flavors	*Percona Server*, *MySQL*, *MariaDB*	*MySQL*
Non-blocking InnoDB backups [1]	Yes	Yes
Blocking MyISAM backups	Yes	Yes
Incremental backups	Yes	Yes
Full compressed backups	Yes	Yes
Incremental compressed backups	Yes	

Continued on next p

[1] *InnoDB* tables are still locked while copying non-*InnoDB* data.

Table 1.1 – continued from previous page

Features	Percona XtraBackup	MySQL Enterprise Backup (InnoDB Hot Bac
Fast incremental backups [2]	Yes	
Incremental backups with archived logs	Yes	
Backup locks [3]	Yes	
Encrypted backups	Yes	Yes [4]
Streaming backups	Yes	Yes
Parallel local backups	Yes	Yes
Parallel streaming backups	Yes	
Parallel compression	Yes	Yes
Parallel encryption	Yes	Yes
Parallel apply-log	Yes	
Parallel copy-back		Yes
Partial backups	Yes	Yes
Throttling [5]	Yes	Yes
Point-in-time recovery support	Yes	Yes
Safe slave backups	Yes	
Compact backups [6]	Yes	Yes
Buffer pool state backups	Yes	
Individual tables export	Yes	Yes [7]
Individual partitions export	Yes	
Restoring tables to a different server [8]	Yes	Yes
Data & index file statistics	Yes	
InnoDB secondary indexes defragmentation	Yes	
`rsync` support to minimize lock time	Yes	
Improved `FTWRL` handling	Yes	
Backup history table	Yes	Yes
Backup progress table		Yes
Offline backups		Yes
Tape backups with Oracle Secure Backup		Yes

1.1.2 What are the features of Percona XtraBackup?

Here is a short list of *Percona XtraBackup* features. See the documentation for more.

- Create hot *InnoDB* backups without pausing your database
- Make incremental backups of *MySQL*
- Stream compressed *MySQL* backups to another server
- Move tables between *MySQL* servers on-line
- Create new *MySQL* replication slaves easily

[2] Fast incremental backups are supported for *Percona Server* with XtraDB changed page tracking enabled.

[3] Backup locks is a lightweight alternative to `FLUSH TABLES WITH READ LOCK` available in *Percona Server* 5.6+. *Percona XtraBackup* uses them automatically to copy non-InnoDB data to avoid blocking DML queries that modify *InnoDB* tables.

[4] *Percona XtraBackup* supports encryption with any kinds of backups. *MySQL Enterprise Backup* only supports encryption for single-file backups.

[5] *Percona XtraBackup* performs throttling based on the number of IO operations per second. *MySQL Enterprise Backup* supports a configurable sleep time between operations.

[6] *Percona XtraBackup* skips secondary index pages and recreates them when a compact backup is prepared. *MySQL Enterprise Backup* skips unused pages and reinserts on the prepare stage.

[7] *Percona XtraBackup* can export individual tables even from a full backup, regardless of the InnoDB version. *MySQL Enterprise Backup* uses InnoDB 5.6 transportable tablespaces only when performing a partial backup.

[8] Tables exported with *Percona XtraBackup* can be imported into *Percona Server* 5.1, 5.5 or 5.6+, or *MySQL* 5.6+. Transportable tablespaces created with *MySQL Enterprise Backup* can only be imported to *Percona Server* 5.6+, *MySQL* 5.6+ or *MariaDB* 10.0+.

- Backup *MySQL* without adding load to the server

CHAPTER

TWO

INSTALLATION

2.1 Installing *Percona XtraBackup* from Binaries

Before installing, you might want to read the *Percona XtraBackup Release Notes*.

Ready-to-use binaries are available from the *Percona XtraBackup* download page, including:

- `RPM` packages for *RHEL* 5 and *RHEL* 6 (including compatible distributions such as CentOS and Oracle Enterprise Linux)
- *Debian* packages for *Debian* and *Ubuntu*
- Generic `.tar.gz` binary packages

2.1.1 Using Percona Software Repositories

Percona apt Repository

Debian and *Ubuntu* packages from *Percona* are signed with a key. Before using the repository, you should add the key to **apt**. To do that, run the following commands:

```
$ apt-key adv --keyserver keys.gnupg.net --recv-keys 1C4CBDCDCD2EFD2A
```

Add this to `/etc/apt/sources.list`, replacing `VERSION` with the name of your distribution:

```
deb http://repo.percona.com/apt VERSION main
deb-src http://repo.percona.com/apt VERSION main
```

Remember to update the local cache:

```
$ apt-get update
```

Supported Architectures

- x86_64 (also known as amd64)
- x86

Supported Releases

Debian

- 6.0 (squeeze)

- 7.0 (wheezy)

Ubuntu

- 10.04LTS (lucid)
- 12.04LTS (precise)
- 13.10 (saucy)
- 14.04LTS (trusty)

Experimental Repository

Percona now offers experimental repository for beta releases. To enable it add this to `/etc/apt/sources.list`, replacing `VERSION` with the name of your distribution:

```
deb http://repo.percona.com/apt VERSION main experimental
deb-src http://repo.percona.com/apt VERSION main experimental
```

Ubuntu PPA of daily builds

Percona offers a Personal Package Archive (PPA) percona-daily/percona-xtrabackup. Every time code is pushed to our main source code repository, the PPA gets updated.

Note: These packages are directly built from lp:percona-xtrabackup/2.1 trunk and should be considered PRE-RELEASE SOFTWARE.

This PPA can be added to your system manually by copying the lines below and adding them to your system's software sources:

```
deb http://ppa.launchpad.net/percona-daily/percona-xtrabackup/ubuntu YOUR_UBUNTU_VERSION_HERE main
deb-src http://ppa.launchpad.net/percona-daily/percona-xtrabackup/ubuntu YOUR_UBUNTU_VERSION_HERE ma
```

or by simply running:

```
$ sudo add-apt-repository ppa:percona-daily/percona-xtrabackup
```

Percona yum Repository

The *Percona* **yum** repository supports popular *RPM*-based operating systems, including the *Amazon Linux AMI*.

The easiest way to install the *Percona Yum* repository is to install an *RPM* that configures **yum** and installs the Percona GPG key. You can also do the installation manually.

Automatic Installation

Execute the following command as a `root` user, replacing `x86_64` with `i386` if you are not running a 64-bit operating system:

```
$ rpm -Uhv http://www.percona.com/downloads/percona-release/percona-release-0.0-1.x86_64.rpm
```

You should see some output such as the following:

```
Retrieving http://www.percona.com/downloads/percona-release/percona-release-0.0-1.x86_64.rpm
Preparing...                ########################################### [100%]
   1:percona-release        ########################################### [100%]
```

The RPMs for the automatic installation are available at http://www.percona.com/downloads/percona-release/ and include source code.

Manual Installation

To install the repository manually, place the following into a new file named `/etc/yum.repos.d/Percona.repo`:

```
[percona]
name = CentOS $releasever - Percona
baseurl=http://repo.percona.com/centos/$releasever/os/$basearch/
enabled = 1
gpgkey = file:///etc/pki/rpm-gpg/RPM-GPG-KEY-percona
gpgcheck = 1
```

Also, copy the Percona GPG key into a file named `/etc/pki/rpm-gpg/RPM-GPG-KEY-percona`.

Testing The Repository

Make sure packages are downloaded from the repository, by executing the following command as root:

```
yum list | grep percona
```

You should see output similar to the following:

```
percona-release.x86_64                  0.0-1                       installed
...
Percona-Server-client-51.x86_64         5.1.47-rel11.1.51.rhel5     percona
Percona-Server-devel-51.x86_64          5.1.47-rel11.1.51.rhel5     percona
Percona-Server-server-51.x86_64         5.1.47-rel11.1.51.rhel5     percona
Percona-Server-shared-51.x86_64         5.1.47-rel11.1.51.rhel5     percona
Percona-Server-test-51.x86_64           5.1.47-rel11.1.51.rhel5     percona
...
xtrabackup.x86_64                       1.2-22.rhel5                percona
```

Supported Platforms

- `x86_64`
- `i386`

Supported Releases

The *CentOS* repositories should work well with *Red Hat Enterprise Linux* too, provided that **yum** is installed on the server.

- *CentOS* 5 and *RHEL* 5
- *CentOS* 6 and *RHEL* 6
- *Amazon Linux AMI* (works the same as *CentOS* 5)

Percona *yum* testing repository

Percona offers fresh beta builds from the experimental repository. To subscribe to the experimental repository, install the experimental *RPM*:

```
yum install http://repo.percona.com/testing/centos/6/os/noarch/percona-testing-0.0-1.noarch.rpm
```

Note: This repository works for both RHEL/CentOS 5 and RHEL/CentOS 6

Percona provides repositories for **yum** (`RPM` packages for *Red Hat Enterprise Linux* and compatible distributions such as *CentOS*, *Oracle Enterprise Linux*, *Amazon Linux AMI*, and *Fedora*) and **apt** (`.deb` packages for *Ubuntu* and *Debian*) for software such as *Percona Server*, *XtraDB*, *Percona XtraBackup*, and *Percona Toolkit*. This makes it easy to install and update your software and its dependencies through your operating system's package manager.

This is the recommend way of installing where possible.

2.2 Compiling and Installing from Source Code

The source code is available from the *Launchpad* project here. The easiest way to get the code is with **bzr branch** of the desired release, such as the following:

```
bzr branch lp:percona-xtrabackup/2.2
```

You should then have a directory named after the release you branched, such as `percona-xtrabackup`.

2.2.1 Compiling on Linux

Prerequisites

The following packages and tools must be installed to compile *Percona XtraBackup* from source. These might vary from system to system.

In Debian-based distributions, you need to:

```
$ apt-get install build-essential flex bison automake autoconf bzr \
   libtool cmake libaio-dev mysql-client libncurses-dev zlib1g-dev \
   libgcrypt11-dev
```

In `RPM`-based distributions, you need to:

```
$ yum install cmake gcc gcc-c++ libaio libaio-devel automake autoconf bzr \
  bison libtool ncurses5-devel
```

Compiling with CMake

At the base directory of the source code tree, if you execute:

```
$ cmake -DBUILD_CONFIG=xtrabackup_release && make -j4
```

and you go for a coffee, at your return *Percona XtraBackup* will be ready to be used.

Installation

The following command:

```
$ make install
```

will install all *Percona XtraBackup* binaries, the **innobackupex** script and tests to `/usr/local/xtrabackup`. You can override this either with:

```
make DESTDIR=... install
```

or by changing the installation layout with:

```
cmake -DINSTALL_LAYOUT=...
```

CHAPTER

THREE

USER'S MANUAL

3.1 *Percona XtraBackup* User Manual

3.1.1 The innobackupex Script

The **innobackupex** tool is a *Perl* script that acts as a wrapper for the *xtrabackup C* program. It is a patched version of the `innobackup` *Perl* script that *Oracle* distributes with the *InnoDB Hot Backup* tool. It enables more functionality by integrating **xtrabackup** and other functions such as file copying and streaming, and adds some convenience. It lets you perform point-in-time backups of *InnoDB* / *XtraDB* tables together with the schema definitions, *MyISAM* tables, and other portions of the server.

We are currently not satisfied with the architecture, code quality and maintainability, or functionality of **innobackupex**, and we expect to replace it with something else in the future.

This manual section explains how to use **innobackupex** in detail.

Prerequisites

Connection and Privileges Needed

Percona XtraBackup needs to be able to connect to the database server and perform operations on the server and the *datadir* when creating a backup, when preparing in some scenarios and when restoring it. In order to do so, there are privileges and permission requirements on its execution that must be fulfilled.

Privileges refers to the operations that a system user is permitted to do in the database server. **They are set at the database server and only apply to users in the database server**.

Permissions are those which permits a user to perform operations on the system, like reading, writing or executing on a certain directory or start/stop a system service. **They are set at a system level and only apply to system users**.

Whether **xtrabackup** or **innobackupex** is used, there are two actors involved: the user invoking the program - *a system user* - and the user performing action in the database server - *a database user*. Note that these are different users in different places, even tough they may have the same username.

All the invocations of **innobackupex** and **xtrabackup** in this documentation assumes that the system user has the appropriate permissions and you are providing the relevant options for connecting the database server - besides the options for the action to be performed - and the database user has adequate privileges.

Connecting to the server The database user used to connect to the server and its password are specified by the `--user` and `--password` option,

```
$ innobackupex --user=DBUSER --password=SECRET /path/to/backup/dir/
$ innobackupex --user=LUKE  --password=US3TH3F0RC3 --stream=tar ./ | bzip2 -
$ xtrabackup --user=DVADER --password=14MY0URF4TH3R --backup --target-dir=/data/bkps/
```

If you don't use the *--user* option, *Percona XtraBackup* will assume the database user whose name is the system user executing it.

Other Connection Options According to your system, you may need to specify one or more of the following options to connect to the server:

Option	Description
–port	The port to use when connecting to the database server with TCP/IP.
–socket	The socket to use when connecting to the local database.
–host	The host to use when connecting to the database server with TCP/IP.

These options are passed to the **mysql** child process without alteration, see *mysql --help* for details.

Note: In case of multiple server instances the correct connection parameters (port, socket, host) must be specified in order for **innobackupex** to talk to the correct server.

Permissions and Privileges Needed Once connected to the server, in order to perform a backup you will need `READ`, `WRITE` and `EXECUTE` permissions at a filesystem level in the server's *datadir*.

The database user needs the following privileges on the tables / databases to be backed up:

- `RELOAD` and `LOCK TABLES` (unless the *--no-lock* option is specified) in order to `FLUSH TABLES WITH READ LOCK` prior to start copying the files and
- `REPLICATION CLIENT` in order to obtain the binary log position,
- `CREATE TABLESPACE` in order to import tables (see *Restoring Individual Tables*) and
- `SUPER` in order to start/stop the slave threads in a replication environment.

The explanation of when these are used can be found in *How innobackupex Works*.

An SQL example of creating a database user with the minimum privileges required to full backups would be:

```
mysql> CREATE USER 'bkpuser'@'localhost' IDENTIFIED BY 's3cret';
mysql> GRANT RELOAD, LOCK TABLES, REPLICATION CLIENT ON *.* TO 'bkpuser'@'localhost';
mysql> FLUSH PRIVILEGES;
```

The Backup Cycle - Full Backups

Creating a Backup with innobackupex

innobackupex is the tool that glues **xtrabackup** and **tar4ibd**, which are specific tools, plus adding functionality to provide a single interface to backup all the data in your database server.

To create a full backup, invoke the script with the options needed to connect to the server and only one argument: the path to the directory where the backup will be stored

```
$ innobackupex --user=DBUSER --password=DBUSERPASS /path/to/BACKUP-DIR/
```

and check the last line of the output for a confirmation message:

```
innobackupex: Backup created in directory '/path/to/BACKUP-DIR/2013-03-25_00-00-09'
innobackupex: MySQL binlog position: filename 'mysql-bin.000003', position 1946
111225 00:00:53  innobackupex: completed OK!
```

The backup will be stored within a time stamped directory created in the provided path, `/path/to/BACKUP-DIR/2013-03-25_00-00-09` in this particular example.

Under the hood **innobackupex** called **xtrabackup** binary to backup all the data of *InnoDB* tables (see *Creating a Backup* for details on this process) and copied all the table definitions in the database (*.frm* files), data and files related to *MyISAM*, *MERGE* (reference to other tables), *CSV* and *ARCHIVE* tables, along with *triggers* and *database configuration information* to a time stamped directory created in the provided path.

It will also create the *following files* for convenience on the created directory.

Other options to consider

The *`--no-timestamp`* option This option tells **innobackupex** not to create a time stamped directory to store the backup:

```
$ innobackupex --user=DBUSER --password=DBUSERPASS /path/to/BACKUP-DIR/ --no-timestamp
```

innobackupex will create the `BACKUP-DIR` subdirectory (or fail if exists) and store the backup inside of it.

The *`--defaults-file`* option You can provide other configuration file to **innobackupex** with this option. The only limitation is that **it has to be the first option passed**:

```
$ innobackupex --defaults-file=/tmp/other-my.cnf --user=DBUSER --password=DBUSERPASS /path/to/BACKUP-
```

Preparing a Full Backup with innobackupex

After creating a backup, the data is not ready to be restored. There might be uncommitted transactions to be undone or transactions in the logs to be replayed. Doing those pending operations will make the data files consistent and it is the purpose of the **prepare stage**. Once this has been done, the data is ready to be used.

To prepare a backup with **innobackupex** you have to use the *`--apply-log`* and the path to the backup directory as an argument:

```
$ innobackupex --apply-log /path/to/BACKUP-DIR
```

and check the last line of the output for a confirmation on the process:

```
111225  1:01:57  InnoDB: Shutdown completed; log sequence number 1609228
111225 01:01:57  innobackupex: completed OK!
```

If it succeeded, **innobackupex** performed all operations needed, leaving the data ready to use immediately.

Under the hood reading the configuration from the files in the backup directory,

innobackupex replayed the committed transactions in the log files (some transactions could have been done while the backup was being done) and rolled back the uncommitted ones. Once this is done, all the information lay in the tablespace (the InnoDB files), and the log files are re-created.

This implies calling **xtrabackup –prepare** twice with the right binary (determined through the `xtrabackup_binary` or by connecting the server). More details of this process are shown in the *xtrabackup section*.

Note that this preparation is not suited for incremental backups. If you perform it on the base of an incremental backup, you will not be able to "add" the increments. See *Incremental Backups with innobackupex*.

Other options to consider

The *`--use-memory`* option The preparing process can be speed up by using more memory in it. It depends on the free or available `RAM` on your system, it defaults to `100MB`. In general, the more memory available to the process, the better. The amount of memory used in the process can be specified by multiples of bytes:

```
$ innobackupex --apply-log --use-memory=4G /path/to/BACKUP-DIR
```

Restoring a Full Backup with innobackupex

For convenience, **innobackupex** has a *`--copy-back`* option, which performs the restoration of a backup to the server's *datadir*

```
$ innobackupex --copy-back /path/to/BACKUP-DIR
```

It will copy all the data-related files back to the server's *datadir*, determined by the server's `my.cnf` configuration file. You should check the last line of the output for a success message:

```
innobackupex: Finished copying back files.

111225 01:08:13  innobackupex: completed OK!
```

Note: The *datadir* must be empty; *Percona XtraBackup* `innobackupex --copy-back` option will not copy over existing files. Also it's important to note that MySQL server needs to be shut down before restore is performed. You can't restore to a *datadir* of a running mysqld instance (except when importing a partial backup).

As files' attributes will be preserved, in most cases you will need to change the files' ownership to `mysql` before starting the database server, as they will be owned by the user who created the backup:

```
$ chown -R mysql:mysql /var/lib/mysql
```

Also note that all of these operations will be done as the user calling **innobackupex**, you will need write permissions on the server's *datadir*.

Other Types of Backups

Incremental Backups with innobackupex

As not all information changes between each backup, the incremental backup strategy uses this to reduce the storage needs and the duration of making a backup.

This can be done because each *InnoDB* page has a log sequence number, *LSN*, which acts as a version number of the entire database. Every time the database is modified, this number gets incremented.

An incremental backup copies all pages since a specific *LSN*.

Once the pages have been put together in their respective order, applying the logs will recreate the process that affected the database, yielding the data at the moment of the most recently created backup.

Creating an Incremental Backups with innobackupex First, you need to make a full backup as the BASE for subsequent incremental backups:

```
$ innobackupex /data/backups
```

This will create a timestamped directory in `/data/backups`. Assuming that the backup is done last day of the month, `BASEDIR` would be `/data/backups/2013-03-31_23-01-18`, for example.

Note: You can use the *`innobackupex --no-timestamp`* option to override this behavior and the backup will be created in the given directory.

If you check at the `xtrabackup-checkpoints` file in `BASE-DIR`, you should see something like:

```
backup_type = full-backuped
from_lsn = 0
to_lsn = 1291135
```

To create an incremental backup the next day, use the *`--incremental`* option and provide the BASEDIR:

```
$ innobackupex --incremental /data/backups --incremental-basedir=BASEDIR
```

and another timestamped directory will be created in `/data/backups`, in this example, `/data/backups/2013-04-01_23-01-18` containing the incremental backup. We will call this `INCREMENTAL-DIR-1`.

If you check at the `xtrabackup-checkpoints` file in `INCREMENTAL-DIR-1`, you should see something like:

```
backup_type = incremental
from_lsn = 1291135
to_lsn = 1352113
```

Creating another incremental backup the next day will be analogous, but this time the previous incremental one will be base:

```
$ innobackupex --incremental /data/backups --incremental-basedir=INCREMENTAL-DIR-1
```

yielding (in this example) `/data/backups/2013-04-02_23-01-18`. We will use `INCREMENTAL-DIR-2` instead for simplicity.

At this point, the `xtrabackup-checkpoints` file in `INCREMENTAL-DIR-2` should contain something like:

```
backup_type = incremental
from_lsn = 1352113
to_lsn = 1358967
```

As it was said before, an incremental backup only copy pages with a *LSN* greater than a specific value. Providing the *LSN* would have produced directories with the same data inside:

```
innobackupex --incremental /data/backups --incremental-lsn=1291135
innobackupex --incremental /data/backups --incremental-lsn=1358967
```

This is a very useful way of doing an incremental backup, since not always the base or the last incremental will be available in the system.

Warning: This procedure only affects *XtraDB* or *InnoDB*-based tables. Other tables with a different storage engine, e.g. *MyISAM*, will be copied entirely each time an incremental backup is performed.

Preparing an Incremental Backup with innobackupex Preparing incremental backups is a bit different than full ones. This is, perhaps, the stage where more attention is needed:

- First, **only the committed transactions must be replayed on each backup**. This will merge the base full backup with the incremental ones.
- Then, the uncommitted transaction must be rolled back in order to have a ready-to-use backup.

If you replay the committed transactions **and** rollback the uncommitted ones on the base backup, you will not be able to add the incremental ones. If you do this on an incremental one, you won't be able to add data from that moment and the remaining increments.

Having this in mind, the procedure is very straight-forward using the *--redo-only* option, starting with the base backup:

```
innobackupex --apply-log --redo-only BASE-DIR
```

You should see an output similar to:

```
120103 22:00:12 InnoDB: Shutdown completed; log sequence number 1291135
120103 22:00:12 innobackupex: completed OK!
```

Then, the first incremental backup can be applied to the base backup, by issuing:

```
innobackupex --apply-log --redo-only BASE-DIR --incremental-dir=INCREMENTAL-DIR-1
```

You should see an output similar to the previous one but with corresponding *LSN*:

```
120103 22:08:43 InnoDB: Shutdown completed; log sequence number 1358967
120103 22:08:43 innobackupex: completed OK!
```

If no *--incremental-dir* is set, **innobackupex** will use the most recent subdirectory created in the basedir.

At this moment, `BASE-DIR` contains the data up to the moment of the first incremental backup. Note that the full data will always be in the directory of the base backup, as we are appending the increments to it.

Repeat the procedure with the second one:

```
innobackupex --apply-log BASE-DIR --incremental-dir=INCREMENTAL-DIR-2
```

If the "completed OK!" message was shown, the final data will be in the base backup directory, `BASE-DIR`.

Note: *--redo-only* should be used when merging all incrementals except the last one. That's why the previous line doesn't contain the *--redo-only* option. Even if the *--redo-only* was used on the last step, backup would still be consistent but in that case server would perform the rollback phase.

You can use this procedure to add more increments to the base, as long as you do it in the chronological order that the backups were done. If you merge the incrementals in the wrong order, the backup will be useless. If you have doubts about the order that they must be applied, you can check the file `xtrabackup_checkpoints` at the directory of each one, as shown in the beginning of this section.

Once you merge the base with all the increments, you can prepare it to roll back the uncommitted transactions:

```
innobackupex --apply-log BASE-DIR
```

Now your backup is ready to be used immediately after restoring it. This preparation step is optional. However, if you restore without doing the prepare, the database server will begin to rollback uncommitted transactions, the same work it would do if a crash had occurred. This results in delay as the database server starts, and you can avoid the delay if you do the prepare.

Note that the `iblog*` files will not be created by **innobackupex**, if you want them to be created, use **xtrabackup –prepare** on the directory. Otherwise, the files will be created by the server once started.

Restoring Incremental Backups with innobackupex After preparing the incremental backups, the base directory contains the same as a full one. For restoring it you can use:

```
innobackupex --copy-back BASE-DIR
```

and you may have to change the ownership as detailed on *Restoring a Full Backup with innobackupex*.

Incremental Streaming Backups using xbstream and tar Incremental streaming backups can be performed with the *xbstream* streaming option. Currently backups are packed in custom **xbstream** format. With this feature taking a BASE backup is needed as well.

Taking a base backup:

```
innobackupex /data/backups
```

Taking a local backup:

```
innobackupex --incremental --incremental-lsn=LSN-number --stream=xbstream ./ > incremental.xbstream
```

Unpacking the backup:

```
xbstream -x < incremental.xbstream
```

Taking a local backup and streaming it to the remote server and unpacking it:

```
innobackupex  --incremental --incremental-lsn=LSN-number --stream=xbstream ./ | /
ssh user@hostname " cat - | xbstream -x -C > /backup-dir/"
```

Partial Backups

Percona XtraBackup features partial backups, which means that you may backup only some specific tables or databases. The tables you back up must be in separate tablespaces, as a result of being created or altered after you enabled the *innodb_file_per_table* option on the server.

There is only one caveat about partial backups: do not copy back the prepared backup. Restoring partial backups should be done by importing the tables, not by using the traditional *--copy-back* option. Although there are some scenarios where restoring can be done by copying back the files, this may be lead to database inconsistencies in many cases and it is not the recommended way to do it.

Creating Partial Backups There are three ways of specifying which part of the whole data will be backed up: regular expressions (*--include*), enumerating the tables in a file (*--tables-file*) or providing a list of databases (*--databases*).

Using the *--include* option The regular expression provided to this will be matched against the fully qualified table name, including the database name, in the form `databasename.tablename`.

For example,

```
$ innobackupex --include='^mydatabase[.]mytable' /path/to/backup
```

The command above will create a timestamped directory with the usual files that **innobackupex** creates, but only the data files related to the tables matched.

Note that this option is passed to *xtrabackup --tables* and is matched against each table of each database, the directories of each database will be created even if they are empty.

Using the *--tables-file* option The text file provided (the path) to this option can contain multiple table names, one per line, in the `databasename.tablename` format.

For example,

```
$ echo "mydatabase.mytable" > /tmp/tables.txt
$ innobackupex --tables-file=/tmp/tables.txt /path/to/backup
```

The command above will create a timestamped directory with the usual files that **innobackupex** creates, but only containing the data-files related to the tables specified in the file.

This option is passed to *xtrabackup --tables-file* and, unlike the *--tables* option, only directories of databases of the selected tables will be created.

Using the *--databases* option This option is specific to **innobackupex** and accepts either a space-separated list of the databases and tables to backup - in the `databasename[.tablename]` form - or a file containing the list at one element per line.

For example,

```
$ innobackupex --databases="mydatabase.mytable mysql" /path/to/backup
```

The command above will create a timestamped directory with the usual files that **innobackupex** creates, but only containing the data-files related to `mytable` in the `mydatabase` directory and the `mysql` directory with the entire `mysql` database.

Note: Currently in *Percona XtraBackup* the –databases option has no effect for InnoDB files for both local and streaming backups, i.e. all InnoDB files are always backed up. Currently, only .frm and non-InnoDB tables are limited by that option.

Preparing Partial Backups For preparing partial backups, the procedure is analogous to *restoring individual tables* : apply the logs and use the *--export* option:

```
$ innobackupex --apply-log --export /path/to/partial/backup
```

You may see warnings in the output about tables that don't exist. This is because *InnoDB* -based engines stores its data dictionary inside the tablespace files besides the *.frm* files. **innobackupex** will use **xtrabackup** to remove the missing tables (those who weren't selected in the partial backup) from the data dictionary in order to avoid future warnings or errors:

```
111225  0:54:06  InnoDB: Error: table 'mydatabase/mytablenotincludedinpartialb'
InnoDB: in InnoDB data dictionary has tablespace id 6,
InnoDB: but tablespace with that id or name does not exist. It will be removed from data dictionary.
```

You should also see the notification of the creation of a file needed for importing (*.exp* file) for each table included in the partial backup:

```
xtrabackup: export option is specified.
xtrabackup: export metadata of table 'employees/departments' to file `.//departments.exp` (2 indexes)
xtrabackup:     name=PRIMARY, id.low=80, page=3
xtrabackup:     name=dept_name, id.low=81, page=4
```

Note that you can use the *--export* option with *--apply-log* to an already-prepared backup in order to create the *.exp* files.

Finally, check the for the confirmation message in the output:

```
111225 00:54:18  innobackupex: completed OK!
```

Restoring Partial Backups Restoring should be done by *restoring individual tables* in the partial backup to the server.

It can also be done by copying back the prepared backup to a "clean" *datadir* (in that case, make sure to include the `mysql` database). System database can be created with:

```
$ sudo mysql_install_db --user=mysql
```

Compact Backups

When doing the backup of *InnoDB* tables it's possible to omit the secondary index pages. This will make the backups more compact and this way they will take less space on disk. The downside of this is that the backup prepare process takes longer as those secondary indexes need to be recreated. Difference in backup size depends on the size of the secondary indexes.

For example full backup taken without and with the *--compact* option:

```
#backup size without --compact
2.0G  2013-02-01_10-18-38

#backup size taken with --compact option
1.4G  2013-02-01_10-29-48
```

Note: Compact backups are not supported for system table space, so in order to work correctly *innodb-file-per-table* option should be enabled.

This feature was introduced in *Percona XtraBackup* 2.1.

Creating Compact Backups To make a compact backup innobackupex needs to be started with the *--compact* option:

```
$ innobackupex --compact /data/backups
```

This will create a timestamped directory in `/data/backups`.

Note: You can use the *innobackupex --no-timestamp* option to override this behavior and the backup will be created in the given directory.

If you check at the `xtrabackup_checkpoints` file in `BASE-DIR`, you should see something like:

```
backup_type = full-backuped
from_lsn = 0
to_lsn = 2888984349
last_lsn = 2888984349
compact = 1
```

When *--compact* wasn't used `compact` value will be `0`. This way it's easy to check if the backup contains the secondary index pages or not.

Preparing Compact Backups Preparing the compact require rebuilding the indexes as well. In order to prepare the backup a new option *--rebuild-indexes* should be used with *--apply-logs*:

```
$ innobackupex --apply-log --rebuild-indexes /data/backups/2013-02-01_10-29-48
```

Output, beside the standard **innobackupex** output, should contain the information about indexes being rebuilt, like:

```
130201 10:40:20  InnoDB: Waiting for the background threads to start
Rebuilding indexes for table sbtest/sbtest1 (space id: 10)
  Found index k_1
  Dropping 1 index(es).
  Rebuilding 1 index(es).
Rebuilding indexes for table sbtest/sbtest2 (space id: 11)
  Found index k_1
  Found index c
  Found index k
  Found index c_2
  Dropping 4 index(es).
  Rebuilding 4 index(es).
```

Since *Percona XtraBackup* has no information when applying an incremental backup to a compact full one, on whether there will be more incremental backups applied to it later or not, rebuilding indexes needs to be explicitly requested by a user whenever a full backup with some incremental backups merged is ready to be restored. Rebuilding indexes unconditionally on every incremental backup merge is not an option, since it is an expensive operation.

Note: To process individual tables in parallel when rebuilding indexes, *innobackupex --rebuild-threads* option can be used to specify the number of threads started by *Percona XtraBackup* when rebuilding secondary indexes on –apply-log –rebuild-indexes. Each thread rebuilds indexes for a single `.ibd` tablespace at a time.

Restoring Compact Backups **innobackupex** has a *--copy-back* option, which performs the restoration of a backup to the server's *datadir*

```
$ innobackupex --copy-back /path/to/BACKUP-DIR
```

It will copy all the data-related files back to the server's *datadir*, determined by the server's `my.cnf` configuration file. You should check the last line of the output for a success message:

```
innobackupex: Finished copying back files.
130201 11:08:13  innobackupex: completed OK!
```

Other Reading

- Feature preview: Compact backups in Percona XtraBackup

Encrypted Backups

Percona XtraBackup has implemented support for encrypted backups. This feature was introduced in *Percona XtraBackup* 2.1. It can be used to encrypt/decrypt local or streaming backup with *xbstream* option (streaming tar backups are not supported) in order to add another layer of protection to the backups. Encryption is done with the `libgcrypt` library.

Note: Encryption related options are currently ignored by **innobackupex** when specified in `my.cnf`.

Creating Encrypted Backups To make an encrypted backup following options need to be specified (options *--encrypt-key* and *--encrypt-key-file* are mutually exclusive, i.e. just one of them needs to be provided):

- *--encryption=ALGORITHM* - currently supported algorithms are: AES128, AES192 and AES256
- *--encrypt-key=ENCRYPTION_KEY* - proper length encryption key to use. It is not recommended to use this option where there is uncontrolled access to the machine as the command line and thus the key can be viewed as part of the process info.
- *--encrypt-key-file=KEYFILE* - the name of a file where the raw key of the appropriate length can be read from. The file must be a simple binary (or text) file that contains exactly the key to be used.

Both *--encrypt-key* option and *--encrypt-key-file* option can be used to specify the encryption key. Encryption key can be generated with command like:

```
$ openssl enc -aes-256-cbc -pass pass:Password -P -md sha1
```

Output of that command should look like this:

```
salt=9464A264486EEC69
key=DDD3A1B6BC90B9A9B631913CF30E0336A2571BA854E2D65CF92A6D0BDBCBB251
iv =A1EDC73815467C083B0869508406637E
```

In this case we can use `iv` value as key.

Using the *--encrypt-key* option Example of the innobackupex command using the *--encrypt-key* should look like this

```
$ innobackupex --encrypt=AES256 --encrypt-key="A1EDC73815467C083B0869508406637E" /data/backups
```

Using the *--encrypt-key-file* option Example of the innobackupex command using the *--encrypt-key-file* should look like this

```
$ innobackupex --encrypt=AES256 --encrypt-key-file=/data/backups/keyfile /data/backups
```

Note: Depending on the text editor used for making the `KEYFILE`, text file in some cases can contain the CRLF and this will cause the key size to grow and thus making it invalid. Suggested way to do this would be to create the file with: `echo -n "A1EDC73815467C083B0869508406637E" > /data/backups/keyfile`

Both of these examples will create a timestamped directory in `/data/backups` containing the encrypted backup.

Note: You can use the *innobackupex --no-timestamp* option to override this behavior and the backup will be created in the given directory.

Optimizing the encryption process Two new options have been introduced with the encrypted backups that can be used to speed up the encryption process. These are *--encrypt-threads* and *--encrypt-chunk-size*. By using the *--encrypt-threads* option multiple threads can be specified to be used for encryption in parallel. Option *--encrypt-chunk-size* can be used to specify the size (in bytes) of the working encryption buffer for each encryption thread (default is 64K).

Decrypting Encrypted Backups Backups can be decrypted with *The xbcrypt binary*. Following one-liner can be used to encrypt the whole folder:

```
$ for i in `find . -iname "*\.xbcrypt"`; do xbcrypt -d --encrypt-key-file=/root/secret_key --encrypt-
```

In *Percona XtraBackup* 2.1.4 new *innobackupex --decrypt* option has been implemented that can be used to decrypt the backups:

```
$ innobackupex --decrypt=AES256 --encrypt-key="A1EDC73815467C083B0869508406637E" /data/backups/2013-0
```

Use of the *innobackupex --decrypt* will remove the original encrypted files and leave the results in the same location.

Note: *innobackupex --parallel* can be used with *innobackupex --decrypt* option to decrypt multiple files simultaneously.

When the files have been decrypted backup can be prepared.

Preparing Encrypted Backups After the backups have been decrypted, they can be prepared the same way as the standard full backups with the *--apply-logs* option:

```
$ innobackupex --apply-log /data/backups/2013-08-01_08-31-35/
```

Restoring Encrypted Backups **innobackupex** has a *--copy-back* option, which performs the restoration of a backup to the server's *datadir*

```
$ innobackupex --copy-back /path/to/BACKUP-DIR
```

It will copy all the data-related files back to the server's *datadir*, determined by the server's `my.cnf` configuration file. You should check the last line of the output for a success message:

```
innobackupex: Finished copying back files.
130801 11:08:13  innobackupex: completed OK!
```

Other Reading

- The Libgcrypt Reference Manual

Advanced Features

Streaming and Compressing Backups

Streaming mode, supported by *Percona XtraBackup*, sends backup to `STDOUT` in special `tar` or *xbstream* format instead of copying files to the backup directory.

This allows you to use other programs to filter the output of the backup, providing greater flexibility for storage of the backup. For example, compression is achieved by piping the output to a compression utility. One of the benefits of streaming backups and using Unix pipes is that the backups can be automatically encrypted.

To use the streaming feature, you must use the *--stream*, providing the format of the stream (`tar` or `xbstream`) and where to store the temporary files:

```
$ innobackupex --stream=tar /tmp
```

innobackupex starts **xtrabackup** in *--log-stream* mode in a child process, and redirects its log to a temporary file. It then uses *xbstream* to stream all of the data files to `STDOUT`, in a special `xbstream` format. See *The xbstream Binary* for details. After it finishes streaming all of the data files to `STDOUT`, it stops xtrabackup and streams the saved log file too.

When compression is enabled, **xtrabackup** compresses all output data, except the meta and non-InnoDB files which are not compressed, using the specified compression algorithm. The only currently supported algorithm is `quicklz`. The resulting files have the qpress archive format, i.e. every *.qp file produced by xtrabackup is essentially a one-file

qpress archive and can be extracted and uncompressed by the qpress file archiver which is available from *Percona Software repositories*.

Using *xbstream* as a stream option, backups can be copied and compressed in parallel which can significantly speed up the backup process. In case backups were both compressed and encrypted, they'll need to decrypted first in order to be uncompressed.

Examples using xbstream Store the complete backup directly to a single file:

```
$ innobackupex --stream=xbstream /root/backup/ > /root/backup/backup.xbstream
```

To stream and compress the backup:

```
$ innobackupex --stream=xbstream --compress /root/backup/ > /root/backup/backup.xbstream
```

To unpack the backup to the /root/backup/ directory:

```
$ xbstream -x <  backup.xbstream -C /root/backup/
```

To send the compressed backup to another host and unpack it:

```
$ innobackupex --compress --stream=xbstream /root/backup/ | ssh user@otherhost "xbstream -x -C /root/
```

Examples using tar Store the complete backup directly to a tar archive:

```
$ innobackupex --stream=tar /root/backup/ > /root/backup/out.tar
```

To send the tar archive to another host:

```
$ innobackupex --stream=tar ./ | ssh user@destination \ "cat - > /data/backups/backup.tar"
```

Warning: To extract *Percona XtraBackup*'s archive you **must** use **tar** with `-i` option:

```
$ tar -xizf backup.tar.gz
```

Compress with your preferred compression tool:

```
$ innobackupex --stream=tar ./ | gzip - > backup.tar.gz
$ innobackupex --stream=tar ./ | bzip2 - > backup.tar.bz2
```

Note that the streamed backup will need to be prepared before restoration. Streaming mode does not prepare the backup.

Taking Backups in Replication Environments

There are options specific to back up from a replication slave.

The `--slave-info` option This option is useful when backing up a replication slave server. It prints the binary log position and name of the master server. It also writes this information to the `xtrabackup_slave_info` file as a `CHANGE MASTER` statement.

This is useful for setting up a new slave for this master can be set up by starting a slave server on this backup and issuing the statement saved in the `xtrabackup_slave_info` file. More details of this procedure can be found in *How to setup a slave for replication in 6 simple steps with Percona XtraBackup*.

The *--safe-slave-backup* option In order to assure a consistent replication state, this option stops the slave SQL thread and wait to start backing up until `Slave_open_temp_tables` in `SHOW STATUS` is zero. If there are no open temporary tables, the backup will take place, otherwise the SQL thread will be started and stopped until there are no open temporary tables. The backup will fail if `Slave_open_temp_tables` does not become zero after *--safe-slave-backup-timeout* seconds (defaults to 300 seconds). The slave SQL thread will be restarted when the backup finishes.

Using this option is always recommended when taking backups from a slave server.

Warning: Make sure your slave is a true replica of the master before using it as a source for backup. A good tool to validate a slave is pt-table-checksum.

Accelerating the backup process

Accelerating with *--parallel* copy and *–compress-threads* When performing a local backup or the streaming backup with *xbstream* option, multiple files can be copied concurrently by using the *--parallel* option. This option specifies the number of threads created by **xtrabackup** to copy data files.

To take advantage of this option either the multiple tablespaces option must be enabled (*innodb_file_per_table*) or the shared tablespace must be stored in multiple *ibdata* files with the *innodb_data_file_path* option. Having multiple files for the database (or splitting one into many) doesn't have a measurable impact on performance.

As this feature is implemented **at a file level**, concurrent file transfer can sometimes increase I/O throughput when doing a backup on highly fragmented data files, due to the overlap of a greater number of random read requests. You should consider tuning the filesystem also to obtain the maximum performance (e.g. checking fragmentation).

If the data is stored on a single file, this option will have no effect.

To use this feature, simply add the option to a local backup, for example:

```
$ innobackupex --parallel=4 /path/to/backup
```

By using the *xbstream* in streaming backups you can additionally speed up the compression process by using the *--compress-threads* option. This option specifies the number of threads created by **xtrabackup** for for parallel data compression. The default value for this option is 1.

To use this feature, simply add the option to a local backup, for example

```
$ innobackupex --stream=xbstream --compress --compress-threads=4 ./ > backup.xbstream
```

Before applying logs, compressed files will need to be uncompressed.

Accelerating with *--rsync* option In order to speed up the backup process and to minimize the time `FLUSH TABLES WITH READ LOCK` is blocking the writes, option *innobackupex --rsync* should be used. When this option is specified, **innobackupex** uses `rsync` to copy all non-InnoDB files instead of spawning a separate `cp` for each file, which can be much faster for servers with a large number of databases or tables. **innobackupex** will call the `rsync` twice, once before the `FLUSH TABLES WITH READ LOCK` and once during to minimize the time the read lock is being held. During the second `rsync` call, it will only synchronize the changes to non-transactional data (if any) since the first call performed before the `FLUSH TABLES WITH READ LOCK`.

Note: This option cannot be used together with *innobackupex --remote-host* or *innobackupex --stream* options.

Throttling backups with innobackupex

Although **innobackupex** does not block your database's operation, any backup can add load to the system being backed up. On systems that do not have much spare I/O capacity, it might be helpful to throttle the rate at which **innobackupex** reads and writes *InnoDB* data. You can do this with the `--throttle` option.

This option is passed directly to **xtrabackup** binary and only limits the operations on the logs and files of *InnoDB* tables. It doesn't have an effect on reading or writing files from tables with other storage engine.

One way of checking the current I/O operations at a system is with **iostat** command. See *Throttling Backups* for details of how throttling works.

The `--throttle` option is similar to the `--sleep` option in `mysqlbackup` and should be used instead of it, as `--sleep` will be ignored.

Restoring Individual Tables

In server versions prior to 5.6, it is not possible to copy tables between servers by copying the files, even with *innodb_file_per_table*. However, with the *Percona XtraBackup*, you can export individual tables from any *InnoDB* database, and import them into *Percona Server* with *XtraDB* or *MySQL* 5.6 (The source doesn't have to be *XtraDB* or or *MySQL* 5.6, but the destination does). This only works on individual *.ibd* files, and cannot export a table that is not contained in its own *.ibd* file.

Note: If you're running *Percona Server* version older than 5.5.10-20.1, variable innodb_expand_import should be used instead of innodb_import_table_from_xtrabackup.

Exporting tables Exporting is done in the preparation stage, not at the moment of creating the backup. Once a full backup is created, prepare it with the `--export` option:

```
$ innobackupex --apply-log --export /path/to/backup
```

This will create for each *InnoDB* with its own tablespace a file with *.exp* extension. An output of this procedure would contain:

```
..
xtrabackup: export option is specified.
xtrabackup: export metadata of table 'mydatabase/mytable' to file
`./mydatabase/mytable.exp` (1 indexes)
..
```

Now you should see a *.exp* file in the target directory:

```
$ find /data/backups/mysql/ -name export_test.*
/data/backups/mysql/test/export_test.exp
/data/backups/mysql/test/export_test.ibd
/data/backups/mysql/test/export_test.cfg
```

These three files are all you need to import the table into a server running *Percona Server* with *XtraDB* or *MySQL* 5.6.

Note: *MySQL* uses `.cfg` file which contains *InnoDB* dictionary dump in special format. This format is different from the `.exp` one which is used in *XtraDB* for the same purpose. Strictly speaking, a `.cfg` file is **not** required to import a tablespace to *MySQL* 5.6 or *Percona Server* 5.6. A tablespace will be imported successfully even if it is from another server, but *InnoDB* will do schema validation if the corresponding `.cfg` file is present in the same directory.

Each *.exp* (or `.cfg`) file will be used for importing that table.

Note: InnoDB does a slow shutdown (i.e. full purge + change buffer merge) on –export, otherwise the tablespaces wouldn't be consistent and thus couldn't be imported. All the usual performance considerations apply: sufficient buffer pool (i.e. –use-memory, 100MB by default) and fast enough storage, otherwise it can take a prohibitive amount of time for export to complete.

Importing tables To import a table to other server, first create a new table with the same structure as the one that will be imported at that server:

```
OTHERSERVER|mysql> CREATE TABLE mytable (...) ENGINE=InnoDB;
```

then discard its tablespace:

```
OTHERSERVER|mysql> ALTER TABLE mydatabase.mytable DISCARD TABLESPACE;
```

After this, copy `mytable.ibd` and `mytable.exp` (or `mytable.cfg` if importing to *MySQL* 5.6) files to database's home, and import its tablespace:

```
OTHERSERVER|mysql> ALTER TABLE mydatabase.mytable IMPORT TABLESPACE;
```

Once this is executed, data in the imported table will be available.

Point-In-Time recovery

Recovering up to particular moment in database's history can be done with **innobackupex** and the binary logs of the server.

Note that the binary log contains the operations that modified the database from a point in the past. You need a full *datadir* as a base, and then you can apply a series of operations from the binary log to make the data match what it was at the point in time you want.

For taking the snapshot, we will use **innobackupex** for a full backup:

```
$ innobackupex /path/to/backup --no-timestamp
```

(the *--no-timestamp* option is for convenience in this example) and we will prepare it to be ready for restoration:

```
$ innobackupex --apply-log /path/to/backup
```

For more details on these procedures, see *Creating a Backup with innobackupex* and *Preparing a Full Backup with innobackupex*.

Now, suppose that time has passed, and you want to restore the database to a certain point in the past, having in mind that there is the constraint of the point where the snapshot was taken.

To find out what is the situation of binary logging in the server, execute the following queries:

```
mysql> SHOW BINARY LOGS;
+------------------+-----------+
| Log_name         | File_size |
+------------------+-----------+
| mysql-bin.000001 |       126 |
| mysql-bin.000002 |      1306 |
| mysql-bin.000003 |       126 |
| mysql-bin.000004 |       497 |
+------------------+-----------+
```

and

```
mysql> SHOW MASTER STATUS;
+------------------+----------+--------------+------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+------------------+----------+--------------+------------------+
| mysql-bin.000004 |      497 |              |                  |
+------------------+----------+--------------+------------------+
```

The first query will tell you which files contain the binary log and the second one which file is currently being used to record changes, and the current position within it. Those files are stored usually in the *datadir* (unless other location is specified when the server is started with the `--log-bin=` option).

To find out the position of the snapshot taken, see the `xtrabackup_binlog_info` at the backup's directory:

```
$ cat /path/to/backup/xtrabackup_binlog_info
mysql-bin.000003      57
```

This will tell you which file was used at moment of the backup for the binary log and its position. That position will be the effective one when you restore the backup:

```
$ innobackupex --copy-back /path/to/backup
```

As the restoration will not affect the binary log files (you may need to adjust file permissions, see *Restoring a Full Backup with innobackupex*), the next step is extracting the queries from the binary log with **mysqlbinlog** starting from the position of the snapshot and redirecting it to a file

```
$ mysqlbinlog /path/to/datadir/mysql-bin.000003 /path/to/datadir/mysql-bin.000004 \
    --start-position=57 > mybinlog.sql
```

Note that if you have multiple files for the binary log, as in the example, you have to extract the queries with one process, as shown above.

Inspect the file with the queries to determine which position or date corresponds to the point-in-time wanted. Once determined, pipe it to the server. Assuming the point is `11-12-25 01:00:00`:

```
$ mysqlbinlog /path/to/datadir/mysql-bin.000003 /path/to/datadir/mysql-bin.000004 \
    --start-position=57 --stop-datetime="11-12-25 01:00:00" | mysql -u root -p
```

and the database will be rolled forward up to that Point-In-Time.

Improved `FLUSH TABLES WITH READ LOCK` handling

When taking backups, `FLUSH TABLES WITH READ LOCK` is being used before the non-InnoDB files are being backed up to ensure backup is being consistent. `FLUSH TABLES WITH READ LOCK` can be run even though there may be a running query that has been executing for hours. In this case everything will be locked up in `Waiting for table flush` or `Waiting for master to send event` states. Killing the `FLUSH TABLES WITH READ LOCK` does not correct this issue either. In this case the only way to get the server operating normally again is to kill off the long running queries that blocked it to begin with. This means that if there are long running queries `FLUSH TABLES WITH READ LOCK` can get stuck, leaving server in read-only mode until waiting for these queries to complete.

In order to prevent this from happening two things have been implemented:

- **innobackupex** can wait for a good moment to issue the global lock.
- **innobackupex** can kill all or only SELECT queries which are preventing the global lock from being acquired

Waiting for queries to finish Good moment to issue a global lock is the moment when there are no long queries running. But waiting for a good moment to issue the global lock for extended period of time isn't always good approach, as it can extend the time needed for backup to take place. To prevent **innobackupex** from waiting to issue `FLUSH TABLES WITH READ LOCK` for too long, new option has been implemented: *`innobackupex --lock-wait-timeout`* option can be used to limit the waiting time. If the good moment to issue the lock did not happen during this time, **innobackupex** will give up and exit with an error message and backup will not be taken. Zero value for this option turns off the feature (which is default).

Another possibility is to specify the type of query to wait on. In this case *`innobackupex --lock-wait-query-type`*. Possible values are `all` and `update`. When `all` is used **innobackupex** will wait for all long running queries (execution time longer than allowed by *`innobackupex --lock-wait-threshold`*) to finish before running the `FLUSH TABLES WITH READ LOCK`. When `update` is used **innobackupex** will wait on `UPDATE/ALTER/REPLACE/INSERT` queries to finish.

Although time needed for specific query to complete is hard to predict, we can assume that queries that are running for a long time already will likely not be completed soon, and queries which are running for a short time will likely be completed shortly. **innobackupex** can use the value of *`innobackupex --lock-wait-threshold`* option to specify which query is long running and will likely block global lock for a while.

Killing the blocking queries Second option is to kill all the queries which prevent global lock from being acquired. In this case all the queries which run longer than `FLUSH TABLES WITH READ LOCK` are possible blockers. Although all queries can be killed, additional time can be specified for the short running queries to complete. This can be specified by *`innobackupex --kill-long-queries-timeout`* option. This option specifies the time for queries to complete, after the value is reached, all the running queries will be killed. Default value is zero, which turns this feature off.

`innobackupex --kill-long-query-type` option can be used to specify all or only `SELECT` queries that are preventing global lock from being acquired. In order to use this option xtrabackup user should have `PROCESS` and `SUPER` privileges.

Options summary

- *`--lock-wait-timeout=N`* (seconds) - how long to wait for a good moment. Default is 0, not to wait.
- *`--lock-wait-query-type={all|update}`* - which long queries should be finished before `FLUSH TABLES WITH READ LOCK` is run. Default is all.
- *`--lock-wait-threshold=N`* (seconds) - how long query should be running before we consider it long running and potential blocker of global lock.
- *`--kill-long-queries-timeout=N`* (seconds) - how many time we give for queries to complete after `FLUSH TABLES WITH READ LOCK` is issued before start to kill. Default if `0`, not to kill.
- *`--kill-long-query-type={all|select}`* - which queries should be killed once *`kill-long-queries-timeout`* has expired.

Example Running the **innobackupex** with the following options:

```
$ innobackupex --lock-wait-threshold=40 --lock-wait-query-type=all --lock-wait-timeout=180 --kill-lon
```

will cause **innobackupex** to spend no longer than 3 minutes waiting for all queries older than 40 seconds to complete. After `FLUSH TABLES WITH READ LOCK` is issued, **innobackupex** will wait 20 seconds for lock to be acquired. If lock is still not acquired after 20 seconds, it will kill all queries which are running longer that the `FLUSH TABLES WITH READ LOCK`.

Version Information This feature has been implemented in *Percona XtraBackup* 2.1.4.

Store backup history on the server

Percona XtraBackup supports storing the backups history on the server. This feature was implemented in *Percona XtraBackup* 2.2. Storing backup history on the server was implemented to provide users with additional information about backups that are being taken. Backup history information will be stored in the *PERCONA_SCHEMA.XTRABACKUP_HISTORY* table.

To use this feature three new **innobackupex** options have been implemented:

- *innobackupex --history* =<name> : This option enables the history feature and allows the user to specify a backup series name that will be placed within the history record.
- *innobackupex --incremental-history-name* =<name> : This option allows an incremental backup to be made based on a specific history series by name. **innobackupex** will search the history table looking for the most recent (highest `to_lsn`) backup in the series and take the `to_lsn` value to use as it's starting lsn. This is mutually exclusive with *innobackupex --incremental-history-uuid*, *innobackupex --incremental-basedir* and *innobackupex --incremental-lsn* options. If no valid LSN can be found (no series by that name) **innobackupex** will return with an error.
- *innobackupex --incremental-history-uuid* =<uuid> : Allows an incremental backup to be made based on a specific history record identified by UUID. **innobackupex** will search the history table looking for the record matching UUID and take the `to_lsn` value to use as it's starting LSN. This options is mutually exclusive with *innobackupex --incremental-basedir*, *innobackupex --incremental-lsn* and *innobackupex --incremental-history-name* options. If no valid LSN can be found (no record by that UUID or missing `to_lsn`), **innobackupex** will return with an error.

Note: Backup that's currently being performed will **NOT** exist in the xtrabackup_history table within the resulting backup set as the record will not be added to that table until after the backup has been taken.

If you want access to backup history outside of your backup set in the case of some catastrophic event, you will need to either perform a `mysqldump`, partial backup or `SELECT` * on the history table after **innobackupex** completes and store the results with you backup set.

Privileges User performing the backup will need following privileges:

- `CREATE` privilege in order to create the *PERCONA_SCHEMA.xtrabackup_history* database and table.
- `INSERT` privilege in order to add history records to the *PERCONA_SCHEMA.xtrabackup_history* table.
- `SELECT` privilege in order to use *innobackupex --incremental-history-name* or *innobackupex --incremental-history-uuid* in order for the feature to look up the `innodb_to_lsn` values in the *PERCONA_SCHEMA.xtrabackup_history* table.

PERCONA_SCHEMA.XTRABACKUP_HISTORY table This table contains the information about the previous server backups. Information about the backups will only be written if the backup was taken with *innobackupex --history* option.

Column Name	Description
uuid	Unique backup id
name	User provided name of backup series. There may be multiple entries with the same name used to identify related backups in a series.
tool_name	Name of tool used to take backup
tool_command	Exact command line given to the tool with –password and –encryption_key obfuscated
tool_version	Version of tool used to take backup
ib-backup_version	Version of the xtrabackup binary used to take backup
server_version	Server version on which backup was taken
start_time	Time at the start of the backup
end_time	Time at the end of the backup
lock_time	Amount of time, in seconds, spent calling and holding locks for `FLUSH TABLES WITH READ LOCK`
binlog_pos	Binlog file and position at end of `FLUSH TABLES WITH READ LOCK`
inn-odb_from_lsn	LSN at beginning of backup which can be used to determine prior backups
inn-odb_to_lsn	LSN at end of backup which can be used as the starting lsn for the next incremental
partial	Is this a partial backup, if `N` that means that it's the full backup
incremental	Is this an incremental backup
format	Description of result format (`file`, `tar`, `xbstream`)
compact	Is this a compact backup
compressed	Is this a compressed backup
encrypted	Is this an encrypted backup

Limitiatons

- *`innobackupex --history`* option must be specified only on the innobackupex command line and not within a configuration file in order to be effective.
- *`innobackupex --incremental-history-name`* and *`innobackupex --incremental-history-uuid`* options must be specified only on the innobackupex command line and not within a configuration file in order to be effective.

Implementation

How innobackupex Works

innobackupex is a script written in Perl that wraps the **xtrabackup** and **tar4ibd** binaries and performs the tasks where the performance and efficiency of *C* program isn't needed. In this way, it provides a convinient and integrated approach to backing up in many common scenarios.

The following describes the rationale behind **innobackupex** actions.

Making a Backup If no mode is specified, **innobackupex** will assume the backup mode.

By default, it starts **xtrabackup** with the *`--suspend-at-end`* option, and lets it copy the InnoDB data files. When **xtrabackup** finishes that, **innobackupex** sees it create the `xtrabackup_suspended_2` file and executes `FLUSH TABLES WITH READ LOCK`. Then it begins copying the rest of the files.

If the *`--ibbackup`* is not supplied, **innobackupex** will try to detect it: if the `xtrabackup_binary` file exists on the backup directory, it reads from it which binary of **xtrabackup** will be used. Otherwise, it will try to connect to

the database server in order to determine it. If the connection can't be established, **xtrabackup** will fail and you must specify it (see *Choosing the Right Binary*).

When the binary is determined, the connection to the database server is checked. This is done by connecting, issuing a query, and closing the connection. If everything goes well, the binary is started as a child process.

If it is not an incremental backup, it connects to the server. It waits for slaves in a replication setup if the option `--safe-slave-backup` is set and will flush all tables with **READ LOCK**, preventing all *MyISAM* tables from writing (unless option `--no-lock` is specified).

Note: Locking is done only for MyISAM and other non-InnoDB tables, and only **after** *Percona XtraBackup* is finished backing up all InnoDB/XtraDB data and logs.

Once this is done, the backup of the files will begin. It will backup *.frm*, *.MRG*, *.MYD*, *.MYI*, *.TRG*, *.TRN*, *.ARM*, *.ARZ*, *.CSM*, *.CSV*, `.par`, and *.opt* files.

When all the files are backed up, it resumes **ibbackup** and wait until it finishes copying the transactions done while the backup was done. Then, the tables are unlocked, the slave is started (if the option `--safe-slave-backup` was used) and the connection with the server is closed. Then, it removes the `xtrabackup_suspended_2` file and permits **xtrabackup** to exit.

It will also create the following files in the directory of the backup:

`xtrabackup_checkpoints` containing the *LSN* and the type of backup;

`xtrabackup_binlog_info` containing the position of the binary log at the moment of backing up;

`xtrabackup_binlog_pos_innodb` containing the position of the binary log at the moment of backing up relative to *InnoDB* transactions;

`xtrabackup_slave_info` containing the MySQL binlog position of the master server in a replication setup via `SHOW SLAVE STATUS` if the `--slave-info` option is passed;

`backup-my.cnf` containing only the `my.cnf` options required for the backup. For example, innodb_data_file_path, innodb_log_files_in_group, innodb_log_file_size, innodb_fast_checksum, innodb_page_size, innodb_log_block_size;

`xtrabackup_binary` containing the binary used for the backup;

`mysql-stderr` containing the `STDERR` of **mysqld** during the process and

`mysql-stdout` containing the `STDOUT` of the server.

Finally, the binary log position will be printed to `STDERR` and **innobackupex** will exit returning 0 if all went OK.

Note that the `STDERR` of **innobackupex** is not written in any file. You will have to redirect it to a file, e.g., `innobackupex OPTIONS 2> backupout.log`.

Restoring a backup To restore a backup with **innobackupex** the `--copy-back` option must be used.

innobackupex will read from the `my.cnf` the variables *datadir*, *innodb_data_home_dir*, *innodb_data_file_path*, *innodb_log_group_home_dir* and check that the directories exist.

It will copy the *MyISAM* tables, indexes, etc. (*.frm*, *.MRG*, *.MYD*, *.MYI*, *.TRG*, *.TRN*, *.ARM*, *.ARZ*, *.CSM*, *.CSV*, `par` and *.opt* files) first, *InnoDB* tables and indexes next and the log files at last. It will preserve file's attributes when copying them, you may have to change the files' ownership to `mysql` before starting the database server, as they will be owned by the user who created the backup.

Alternatively, the `--move-back` option may be used to restore a backup. This option is similar to `--copy-back` with the only difference that instead of copying files it moves them to their target locations. As this option removes backup files, it must be used with caution. It is useful in cases when there is not enough free disk space to hold both data files and their backup copies.

References

The innobackupex Option Reference

This page documents all of the command-line options for the **innobackupex** Perl script.

Options

--apply-log
Prepare a backup in BACKUP-DIR by applying the transaction log file named xtrabackup_logfile located in the same directory. Also, create new transaction logs. The InnoDB configuration is read from the file backup-my.cnf created by **innobackupex** when the backup was made. innobackupex –apply-log uses InnoDB configuration from backup-my.cnf by default, or from –defaults-file, if specified. InnoDB configuration in this context means server variables that affect data format, i.e. *innodb_page_size*, *innodb_log_block_size*, etc. Location-related variables, like *innodb_log_group_home_dir* or *innodb_data_file_path* are always ignored by –apply-log, so preparing a backup always works with data files from the backup directory, rather than any external ones.

--compact
Create a compact backup with all secondary index pages omitted. This option is passed directly to xtrabackup. See the **xtrabackup** *documentation* for details.

--compress
This option instructs xtrabackup to compress backup copies of InnoDB data files. It is passed directly to the xtrabackup child process. See the **xtrabackup** *documentation* for details.

--compress-threads=#
This option specifies the number of worker threads that will be used for parallel compression. It is passed directly to the xtrabackup child process. See the **xtrabackup** *documentation* for details.

--compress-chunk-size=#
This option specifies the size of the internal working buffer for each compression thread, measured in bytes. It is passed directly to the xtrabackup child process. The default value is 64K. See the **xtrabackup** *documentation* for details.

--copy-back
Copy all the files in a previously made backup from the backup directory to their original locations.

--databases=LIST
This option specifies the list of databases that **innobackupex** should back up. The option accepts a string argument or path to file that contains the list of databases to back up. The list is of the form "database-name1[.table_name1] databasename2[.table_name2] . . .". If this option is not specified, all databases containing *MyISAM* and *InnoDB* tables will be backed up. Please make sure that –databases contains all of the *InnoDB* databases and tables, so that all of the innodb.frm files are also backed up. In case the list is very long, this can be specified in a file, and the full path of the file can be specified instead of the list. (See option –tables-file.)

--decompress
Decompresses all files with the .qp extension in a backup previously made with the –compress option. The *innobackupex --parallel* option will allow multiple files to be decrypted and/or decompressed simultaneously. In order to decompress, the qpress utility MUST be installed and accessable within the path. This process will remove the original compressed/encrypted files and leave the results in the same location.

--decrypt=ENCRYPTION-ALGORITHM
Decrypts all files with the .xbcrypt extension in a backup previously made with –encrypt option. The *innobackupex --parallel* option will allow multiple files to be decrypted and/or decompressed simultaneously.

--defaults-file=[MY.CNF]
This option accepts a string argument that specifies what file to read the default MySQL options from. It is also passed directly to **xtrabackup** 's defaults-file option. See the **xtrabackup** *documentation* for details.

--defaults-extra-file=[MY.CNF]
This option specifies what extra file to read the default *MySQL* options from before the standard defaults-file. The option accepts a string argument. It is also passed directly to xtrabackup's –defaults-extra-file option. See the **xtrabackup** *documentation* for details.

--defaults-group=GROUP-NAME
This option accepts a string argument that specifies the group which should be read from the configuration file. This is needed if you use mysqld_multi. This can also be used to indicate groups other than mysqld and xtrabackup.

--encrypt=ENCRYPTION_ALGORITHM
This option instructs xtrabackup to encrypt backup copies of InnoDB data files using the algorithm specified in the ENCRYPTION_ALGORITHM. It is passed directly to the xtrabackup child process. See the **xtrabackup** *documentation* for more details.

--encrypt-key=ENCRYPTION_KEY
This option instructs xtrabackup to use the given ENCRYPTION_KEY when using the –encrypt option. It is passed directly to the xtrabackup child process. See the **xtrabackup** *documentation* for more details.

--encrypt-key-file=ENCRYPTION_KEY_FILE
This option instructs xtrabackup to use the encryption key stored in the given ENCRYPTION_KEY_FILE when using the –encrypt option. It is passed directly to the xtrabackup child process. See the **xtrabackup** *documentation* for more details.

--encrypt-threads=#
This option specifies the number of worker threads that will be used for parallel encryption. It is passed directly to the xtrabackup child process. See the **xtrabackup** *documentation* for more details.

--encrypt-chunk-size=#
This option specifies the size of the internal working buffer for each encryption thread, measured in bytes. It is passed directly to the xtrabackup child process. See the **xtrabackup** *documentation* for more details.

--export
This option is passed directly to *xtrabackup --export* option. It enables exporting individual tables for import into another server. See the **xtrabackup** documentation for details.

--extra-lsndir=DIRECTORY
This option accepts a string argument that specifies the directory in which to save an extra copy of the xtrabackup_checkpoints file. It is passed directly to **xtrabackup**'s *--extra-lsndir* option. See the **xtrabackup** documentation for details.

--force-non-empty-directories
When specified, it makes *innobackupex --copy-back* option or *innobackupex --move-back* option transfer files to non-empty directories. No existing files will be overwritten. If –copy-back or –move-back has to copy a file from the backup directory which already exists in the destination directory, it will still fail with an error.

--galera-info
This options creates the xtrabackup_galera_info file which contains the local node state at the time of the backup. Option should be used when performing the backup of Percona-XtraDB-Cluster.

--help
This option displays a help screen and exits.

--history=NAME
This option enables the tracking of backup history in the PERCONA_SCHEMA.xtrabackup_history table.

An optional history series name may be specified that will be placed with the history record for the current backup being taken.

--host=HOST
This option accepts a string argument that specifies the host to use when connecting to the database server with TCP/IP. It is passed to the mysql child process without alteration. See **mysql –help** for details.

--ibbackup=IBBACKUP-BINARY
This option accepts a string argument that specifies which **xtrabackup** binary should be used. The string should be the command used to run *Percona XtraBackup*. The option can be useful if the **xtrabackup** binary is not in your search path or working directory and the database server is not accessible at the moment. If this option is not specified, **innobackupex** attempts to determine the binary to use automatically. By default, **xtrabackup** is the command used. When option *--apply-log* is specified, the binary is used whose name is in the file `xtrabackup_binary` in the backup directory, if that file exists, or will attempt to autodetect it. However, if *--copy-back* or *--move-back* is used, **xtrabackup** is used unless other is specified.

--include=REGEXP
This option is a regular expression to be matched against table names in `databasename.tablename` format. It is passed directly to xtrabackup's *xtrabackup --tables* option. See the **xtrabackup** documentation for details.

--incremental
This option tells **xtrabackup** to create an incremental backup, rather than a full one. It is passed to the **xtrabackup** child process. When this option is specified, either *--incremental-lsn* or *--incremental-basedir* can also be given. If neither option is given, option *--incremental-basedir* is passed to **xtrabackup** by default, set to the first timestamped backup directory in the backup base directory.

--incremental-basedir=DIRECTORY
This option accepts a string argument that specifies the directory containing the full backup that is the base dataset for the incremental backup. It is used with the *--incremental* option.

--incremental-dir=DIRECTORY
This option accepts a string argument that specifies the directory where the incremental backup will be combined with the full backup to make a new full backup. It is used with the *--incremental* option.

--incremental-history-name=NAME
This option specifies the name of the backup series stored in the *PERCONA_SCHEMA.xtrabackup_history* history record to base an incremental backup on. Percona Xtrabackup will search the history table looking for the most recent (highest innodb_to_lsn), successful backup in the series and take the to_lsn value to use as the starting lsn for the incremental backup. This will be mutually exclusive with *innobackupex --incremental-history-uuid*,:option:*innobackupex –incremental-basedir* and *innobackupex --incremental-lsn*. If no valid lsn can be found (no series by that name, no successful backups by that name) xtrabackup will return with an error. It is used with the *innobackupex --incremental* option.

--incremental-history-uuid=UUID
This option specifies the UUID of the specific history record stored in the *PERCONA_SCHEMA.xtrabackup_history* to base an incremental backup on. *innobackupex --incremental-history-name*,:optionL‘innobackupex –incremental-basedir‘ and *innobackupex --incremental-lsn*. If no valid lsn can be found (no success record with that uuid) xtrabackup will return with an error. It is used with the *innobackupex --incremental* option.

--incremental-lsn=LSN
This option accepts a string argument that specifies the log sequence number (*LSN*) to use for the incremental backup. It is used with the *--incremental* option. It is used instead of specifying *--incremental-basedir*. For databases created by *MySQL* and *Percona Server* 5.0-series versions, specify the as two 32-bit integers in high:low format. For databases created in 5.1 and later, specify the LSN as a

single 64-bit integer.

--kill-long-queries-timeout=SECONDS
This option specifies the number of seconds innobackupex waits between starting FLUSH TABLES WITH READ LOCK and killing those queries that block it. Default is 0 seconds, which means innobackupex will not attempt to kill any queries. In order to use this option xtrabackup user should have PROCESS and SUPER privileges.

--kill-long-query-type=all|select
This option specifies which types of queries should be killed to unblock the global lock. Default is "all".

--lock-wait-timeout=SECONDS
This option specifies time in seconds that innobackupex should wait for queries that would block FLUSH TABLES WITH READ LOCK before running it. If there are still such queries when the timeout expires, innobackupex terminates with an error. Default is 0, in which case innobackupex does not wait for queries to complete and starts FLUSH TABLES WITH READ LOCK immediately.

--lock-wait-threshold=SECONDS
This option specifies the query run time threshold which is used by innobackupex to detect long-running queries with a non-zero value of *innobackupex --lock-wait-timeout*. FLUSH TABLES WITH READ LOCK'' is not started until such long-running queries exist. This option has no effect if –lock-wait-timeout is 0. Default value is 60 seconds.

--lock-wait-query-type=all|update
This option specifies which types of queries are allowed to complete before innobackupex will issue the global lock. Default is all.

--log-copy-interval=#
This option specifies time interval between checks done by log copying thread in milliseconds.

--move-back
Move all the files in a previously made backup from the backup directory to their original locations. As this option removes backup files, it must be used with caution.

--no-lock
Use this option to disable table lock with FLUSH TABLES WITH READ LOCK. Use it only if ALL your tables are InnoDB and you **DO NOT CARE** about the binary log position of the backup. This option shouldn't be used if there are any DDL statements being executed or if any updates are happening on non-InnoDB tables (this includes the system MyISAM tables in the *mysql* database), otherwise it could lead to an inconsistent backup. If you are considering to use *--no-lock* because your backups are failing to acquire the lock, this could be because of incoming replication events preventing the lock from succeeding. Please try using *--safe-slave-backup* to momentarily stop the replication slave thread, this may help the backup to succeed and you then don't need to resort to using this option. xtrabackup_binlog_info is not created when –no-lock option is used (because SHOW MASTER STATUS may be inconsistent), but under certain conditions xtrabackup_binlog_pos_innodb can be used instead to get consistent binlog coordinates as described in *Working with Binary Logs*.

--no-timestamp
This option prevents creation of a time-stamped subdirectory of the BACKUP-ROOT-DIR given on the command line. When it is specified, the backup is done in BACKUP-ROOT-DIR instead.

--no-version-check
This option disables the version check which is enabled by the –version-check option.

--parallel=NUMBER-OF-THREADS
This option accepts an integer argument that specifies the number of threads the **xtrabackup** child process should use to back up files concurrently. Note that this option works on file level, that is, if you have several .ibd files, they will be copied in parallel. If your tables are stored together in a single tablespace file, it will have no effect. This option will allow multiple files to be decrypted and/or decompressed simultaneously. In

order to decompress, the qpress utility MUST be installed and accessable within the path. This process will remove the original compressed/encrypted files and leave the results in the same location. It is passed directly to xtrabackup's *xtrabackup --parallel* option. See the **xtrabackup** documentation for details

--password=PASSWORD
This option accepts a string argument specifying the password to use when connecting to the database. It is passed to the **mysql** child process without alteration. See **mysql –help** for details.

--port=PORT
This option accepts a string argument that specifies the port to use when connecting to the database server with TCP/IP. It is passed to the **mysql** child process. It is passed to the **mysql** child process without alteration. See **mysql –help** for details.

--rebuild-indexes
This option only has effect when used together with the –apply-log option and is passed directly to xtrabackup. When used, makes xtrabackup rebuild all secondary indexes after applying the log. This option is normally used to prepare compact backups. See the **xtrabackup** documentation for more information.

--rebuild-threads=NUMBER-OF-THREADS
This option only has effect when used together with the –apply-log and –rebuild-indexes option and is passed directly to xtrabackup. When used, xtrabackup processes tablespaces in parallel with the specified number of threads when rebuilding indexes. See the **xtrabackup** documentation for more information.

--redo-only
This option should be used when preparing the base full backup and when merging all incrementals except the last one. It is passed directly to xtrabackup's *xtrabackup --apply-log-only* option. This forces **xtrabackup** to skip the "rollback" phase and do a "redo" only. This is necessary if the backup will have incremental changes applied to it later. See the **xtrabackup** *documentation* for details.

--rsync
Uses the **rsync** utility to optimize local file transfers. When this option is specified, **innobackupex** uses **rsync** to copy all non-InnoDB files instead of spawning a separate **cp** for each file, which can be much faster for servers with a large number of databases or tables. This option cannot be used together with *--stream*.

--safe-slave-backup
Stop slave SQL thread and wait to start backup until `Slave_open_temp_tables` in `SHOW STATUS` is zero. If there are no open temporary tables, the backup will take place, otherwise the SQL thread will be started and stopped until there are no open temporary tables. The backup will fail if `Slave_open_temp_tables` does not become zero after *--safe-slave-backup-timeout* seconds. The slave SQL thread will be restarted when the backup finishes.

--safe-slave-backup-timeout=SECONDS
How many seconds *--safe-slave-backup* ' should wait for `Slave_open_temp_tables` to become zero. Defaults to 300 seconds.

--scpopt = SCP-OPTIONS
This option accepts a string argument that specifies the command line options to pass to **scp** when the option *--remost-host* is specified. If the option is not specified, the default options are `-Cp -c arcfour`.

--slave-info
This option is useful when backing up a replication slave server. It prints the binary log position and name of the master server. It also writes this information to the `xtrabackup_slave_info` file as a `CHANGE MASTER` command. A new slave for this master can be set up by starting a slave server on this backup and issuing a `CHANGE MASTER` command with the binary log position saved in the `xtrabackup_slave_info` file.

--socket
This option accepts a string argument that specifies the socket to use when connecting to the local database server with a UNIX domain socket. It is passed to the mysql child process without alteration. See **mysql –help** for details.

--sshopt=SSH-OPTIONS
This option accepts a string argument that specifies the command line options to pass to **ssh** when the option *--remost-host* is specified.

--stream=STREAMNAME
This option accepts a string argument that specifies the format in which to do the streamed backup. The backup will be done to STDOUT in the specified format. Currently, supported formats are *tar* and *xbstream*. Uses *xbstream*, which is available in *Percona XtraBackup* distributions. If you specify a path after this option, it will be interpreted as the value of *tmpdir*

--tables-file=FILE
This option accepts a string argument that specifies the file in which there are a list of names of the form database.table, one per line. The option is passed directly to **xtrabackup** 's *--tables-file* option.

--throttle=IOS
This option accepts an integer argument that specifies the number of I/O operations (i.e., pairs of read+write) per second. It is passed directly to xtrabackup's *xtrabackup --throttle* option.

--tmpdir=DIRECTORY
This option accepts a string argument that specifies the location where a temporary file will be stored. It may be used when *--stream* is specified. For these options, the transaction log will first be stored to a temporary file, before streaming or copying to a remote host. This option specifies the location where that temporary file will be stored. If the option is not specified, the default is to use the value of tmpdir read from the server configuration. innobackupex is passing the tmpdir value specified in my.cnf as the –target-dir option to the xtrabackup binary. Both [mysqld] and [xtrabackup] groups are read from my.cnf. If there is tmpdir in both, then the value being used depends on the order of those group in my.cnf.

--use-memory=#
This option accepts a string argument that specifies the amount of memory in bytes for **xtrabackup** to use for crash recovery while preparing a backup. Multiples are supported providing the unit (e.g. 1MB, 1M, 1GB, 1G). It is used only with the option *--apply-log*. It is passed directly to xtrabackup's *xtrabackup --use-memory* option. See the **xtrabackup** documentation for details.

--user=USER
This option accepts a string argument that specifies the user (i.e., the *MySQL* username used when connecting to the server) to login as, if that's not the current user. It is passed to the mysql child process without alteration. See **mysql –help** for details.

--version
This option displays the **innobackupex** version and copyright notice and then exits.

--version-check
When this option is specified, innobackupex will perform a version check against the server on the backup stage after creating a server connection.

3.1.2 The xtrabackup Binary

The **xtrabackup** binary is a compiled C program that is linked with the *InnoDB* libraries and the standard *MySQL* client libraries. The *InnoDB* libraries provide functionality necessary to apply a log to data files, and the *MySQL* client libraries provide command-line option parsing, configuration file parsing, and so on to give the binary a familiar look and feel.

The tool runs in either *--backup* or *--prepare* mode, corresponding to the two main functions it performs. There are several variations on these functions to accomplish different tasks, and there are two less commonly used modes, *--stats* and *--print-param*.

Getting Started with xtrabackup

Choosing the Right Binary

The **xtrabackup** binary actually exists as four separate binaries: `xtrabackup`, `xtrabackup_51`, `xtrabackup_55` and `xtrabackup_56`. This is to ensure binary compatibility of *InnoDB* data files across releases. Which binary you should use depends on the version of the server that produced the data files you want to back up. It's important to use the correct binary. Whichever binary is used to *create a backup* should also be the binary used *to prepare that backup*.

Throughout the documentation, whenever the **xtrabackup** binary is mentioned, it's assumed that the name of the correct binary will be substituted. The following table summarizes which binary should be used with different server versions.

Server	**xtrabackup** binary
MySQL 5.1.*	`xtrabackup_51`
MySQL 5.1.* with InnoDB plugin	`xtrabackup`
MySQL 5.5.*	`xtrabackup_55`
MySQL 5.6.*	`xtrabackup_56`
MariaDB 5.1.*	`xtrabackup`
MariaDB 5.2.*	`xtrabackup`
MariaDB 5.3.*	`xtrabackup`
MariaDB 5.5.*	`xtrabackup_55`
MariaDB 10.0.*	`xtrabackup_56`
Percona Server 5.0	`xtrabackup_51`
Percona Server 5.1	`xtrabackup`
Percona Server 5.5	`xtrabackup_55`
Percona Server 5.6	`xtrabackup_56`

Configuring xtrabackup

All of the **xtrabackup** configuration is done through options, which behave exactly like standard *MySQL* program options: they can be specified either at the command-line, or through a file such as `/etc/my.cnf`.

The **xtrabackup** binary reads the `[mysqld]` and `[xtrabackup]` sections from any configuration files, in that order. That is so that it can read its options from your existing *MySQL* installation, such as the *datadir* or some of the *InnoDB* options. If you want to override these, just specify them in the `[xtrabackup]` section, and because it is read later, it will take precedence.

You don't need to put any configuration in your `my.cnf` if you don't want to. You can simply specify the options on the command-line. Normally, the only thing you might find convenient to place in the `[xtrabackup]` section of your `my.cnf` file is the `target_dir` option to default the directory in which the backups will be placed, for example,

```
[xtrabackup]
target_dir = /data/backups/mysql/
```

This manual will assume that you do not have any file-based configuration for **xtrabackup**, so it will always show command-line options being used explicitly. Please see the *option and variable reference* for details on all of the configuration options.

The **xtrabackup** binary does not accept exactly the same syntax in the `my.cnf` file as the **mysqld** server binary does. For historical reasons, the **mysqld** server binary accepts parameters with a `--set-variable=<variable>=<value>` syntax, which **xtrabackup** does not understand. If your `my.cnf` file has such configuration directives, you should rewrite them in the `--variable=value` syntax.

System Configuration and NFS Volumes The **xtrabackup** tool requires no special configuration on most systems. However, the storage where the `--target-dir` is located must behave properly when `fsync()` is called. In particular, we have noticed that NFS volumes not mounted with the `sync` option might not really sync the data. As a result, if you back up to an NFS volume mounted with the async option, and then try to prepare the backup from a different server that also mounts that volume, the data might appear to be corrupt. You can use the `sync` mount option to avoid this problem.

The Backup Cycle - Full Backups

Creating a Backup

To create a backup, run **xtrabackup** with the **–backup** option. You also need to specify a `--target_dir` option, which is where the backup will be stored, and a `--datadir` option, which is where the *MySQL* data is stored. If the *InnoDB* data or log files aren't stored in the same directory, you might need to specify the location of those, too. If the target directory does not exist, **xtrabackup** creates it. If the directory does exist and is empty, **xtrabackup** will succeed. **xtrabackup** will not overwrite existing files, it will fail with operating system error 17, `file exists`.

The tool changes its working directory to the data directory and performs two primary tasks to complete the backup:

- It starts a log-copying thread in the background. This thread watches the InnoDB log files, and when they change, it copies the changed blocks to a file called `xtrabackup_logfile` in the backup target directory. This is necessary because the backup might take a long time, and the recovery process needs all of the log file entries from the beginning to the end of the backup.
- It copies the *InnoDB* data files to the target directory. This is not a simple file copy; it opens and reads the files similarly to the way *InnoDB* does, by reading the data dictionary and copying them a page at a time.

When the data files are finished copying, **xtrabackup** stops the log-copying thread, and creates a files in the target directory called `xtrabackup_checkpoints`, which contains the type of backup performed, the log sequence number at the beginning, and the log sequence number at the end.

An example command to perform a backup follows:

```
$ xtrabackup --backup --datadir=/var/lib/mysql/ --target-dir=/data/backups/mysql/
```

This takes a backup of `/var/lib/mysql` and stores it at `/data/backups/mysql/`. If you specify a relative path, the target directory will be relative to the current directory.

During the backup process, you should see a lot of output showing the data files being copied, as well as the log file thread repeatedly scanning the log files and copying from it. Here is an example that shows the log thread scanning the log in the background, and a file copying thread working on the `ibdata1` file:

```
>> log scanned up to (3646475465483)
>> log scanned up to (3646475517369)
>> log scanned up to (3646475581716)
>> log scanned up to (3646475636841)
>> log scanned up to (3646475718082)
>> log scanned up to (3646475988095)
>> log scanned up to (3646476048286)
>> log scanned up to (3646476102877)
>> log scanned up to (3646476140854)
[01] Copying /usr/local/mysql/var/ibdata1
     to /usr/local/mysql/Backups/2011-04-18_21-11-15/ibdata1
[01]        ...done
```

The last thing you should see is something like the following, where the value of the `<LSN>` will be a number that depends on your system:

```
xtrabackup: Transaction log of lsn (<SLN>) to (<LSN>) was copied.
```

Note: Log copying thread checks the transactional log every second to see if there were any new log records written that need to be copied, but there is a chance that the log copying thread might not be able to keep up with the amount of writes that go to the transactional logs, and will hit an error when the log records are overwritten before they could be read.

After the backup is finished, the target directory will contain files such as the following, assuming you have a single InnoDB table `test.tbl1` and you are using MySQL's *innodb_file_per_table* option:

```
/data/backups/mysql/ibdata1
/data/backups/mysql/test
/data/backups/mysql/test/tbl1.ibd
/data/backups/mysql/xtrabackup_checkpoints
/data/backups/mysql/xtrabackup_logfile
```

The backup can take a long time, depending on how large the database is. It is safe to cancel at any time, because it does not modify the database.

The next step is getting your backup ready to restored: *Preparing the backup*.

Preparing the backup

After you make a backup with *--backup*, the next step is to prepare it. The data files are not point-in-time consistent until they've been prepared, because they were copied at different times as the program ran, and they might have been changed while this was happening. If you try to start InnoDB with these data files, it will detect corruption and crash itself to prevent you from running on damaged data. The *--prepare* step makes the files perfectly consistent at a single instant in time, so you can run *InnoDB* on them.

Note: For prepare "innobackupex –apply-log" should be used which will read *InnoDB* configuration from backup-my.cnf automatically, or –defaults-file=backup-my.cnf should be passed to the xtrabackup binary if it is used for preparing the backup. Otherwise it could lead to incorrect restore because xtrabackup could use wrong configuration options.

You can run the prepare operation on any machine; it does not need to be on the originating server or the server to which you intend to restore. You can copy the backup to a utility server and prepare it there, for example. It is important, however, that you use the same version of the xtrabackup binary that you used to create the backup to do the prepare.

During the prepare operation, **xtrabackup** boots up a kind of modified InnoDB that's embedded inside it (the libraries it was linked against). The modifications are necessary to disable InnoDB's standard safety checks, such as complaining that the log file isn't the right size, which aren't appropriate for working with backups. These modifications are only for the xtrabackup binary; you don't need a modified *InnoDB* to use **xtrabackup** for your backups.

The prepare step uses this "embedded InnoDB" to perform crash recovery on the copied datafiles, using the copied log file. The `prepare` step is very simple to use: you simply run **xtrabackup** with the *--prepare* option and tell it which directory to prepare, for example, to prepare the backup previously taken,

```
xtrabackup --prepare --target-dir=/data/backups/mysql/
```

When this finishes, you should see an "InnoDB shutdown" with a message such as the following, where again the value of *LSN* will depend on your system:

```
101107 16:40:15  InnoDB: Shutdown completed; log sequence number <LSN>
```

Your backup is now clean and consistent, and ready to restore. However, you might want to take an extra step to make restores as quick as possible. This is to prepare the backup a second time. The first time makes the data files perfectly self-consistent, but it doesn't create fresh *InnoDB* log files. If you restore the backup at this point and start *MySQL*, it will have to create new log files, which could take a little while, and you might not want to wait for that. If you run `--prepare` a second time, **xtrabackup** will create the log files for you, and output status text such as the following, which is abbreviated for clarity. The value of `<SIZE>` will depend on your MySQL configuration.

```
$ xtrabackup --prepare --target-dir=/data/backups/mysql/
xtrabackup: This target seems to be already prepared.
xtrabackup: notice: xtrabackup_logfile was already used to '--prepare'.
101107 16:54:10  InnoDB: Log file ./ib_logfile0 did not exist: new to be created
InnoDB: Setting log file ./ib_logfile0 size to <SIZE> MB
InnoDB: Database physically writes the file full: wait...
101107 16:54:10  InnoDB: Log file ./ib_logfile1 did not exist: new to be created
InnoDB: Setting log file ./ib_logfile1 size to <SIZE> MB
InnoDB: Database physically writes the file full: wait...
101107 16:54:15  InnoDB: Shutdown completed; log sequence number 1284108
```

All following prepares (third and following) will not change the already prepared data files, you can only see that output says

```
xtrabackup: This target seems to be already prepared.
xtrabackup: notice: xtrabackup_logfile was already used to '--prepare'.
```

It is not recommended to interrupt xtrabackup process while preparing backup - it may cause data files corruption and backup will become not usable. Backup validity is not guaranteed if prepare process was interrupted.

If you intend the backup to be the basis for further incremental backups, you should use the `--apply-log-only` option when preparing the backup, or you will not be able to apply incremental backups to it. See the documentation on preparing *incremental backups* for more details.

Restoring a Backup

The **xtrabackup** binary does not have any functionality for restoring a backup. That is up to the user to do. You might use **rsync** or **cp** to restore the files. You should check that the restored files have the correct ownership and permissions.

Note: The *datadir* must be empty before restoring the backup. Also it's important to note that MySQL server needs to be shut down before restore is performed. You can't restore to a *datadir* of a running mysqld instance (except when importing a partial backup).

Example of the **rsync** command that can be used to restore the backup can look like this:

```
$ rsync -avrP /data/backup/ /var/lib/mysql/
```

As files' attributes will be preserved, in most cases you will need to change the files' ownership to `mysql` before starting the database server, as they will be owned by the user who created the backup:

```
$ chown -R mysql:mysql /var/lib/mysql
```

Note that **xtrabackup** backs up only the *InnoDB* data. You must separately restore the *MySQL* system database, *MyISAM* data, table definition files (*.frm* files), and everything else necessary to make your database functional – or **innobackupex** *can do it for you*.

Other Types of Backups

Incremental Backups

Both **xtrabackup** and **innobackupex** tools supports incremental backups, which means that it can copy only the data that has changed since the last full backup. You can perform many incremental backups between each full backup, so you can set up a backup process such as a full backup once a week and an incremental backup every day, or full backups every day and incremental backups every hour.

Incremental backups work because each InnoDB page (usually 16kb in size) contains a log sequence number, or *LSN*. The *LSN* is the system version number for the entire database. Each page's *LSN* shows how recently it was changed. An incremental backup copies each page whose *LSN* is newer than the previous incremental or full backup's *LSN*. There are two algorithms in use to find the set of such pages to be copied. The first one, available with all the server types and versions, is to check the page *LSN* directly by reading all the data pages. The second one, available with *Percona Server*, is to enable the changed page tracking feature on the server, which will note the pages as they are being changed. This information will be then written out in a compact separate so-called bitmap file. The **xtrabackup** binary will use that file to read only the data pages it needs for the incremental backup, potentially saving many read requests. The latter algorithm is enabled by default if the **xtrabackup** binary finds the bitmap file. It is possible to specify `--incremental-force-scan` to read all the pages even if the bitmap data is available.

Incremental backups do not actually compare the data files to the previous backup's data files. In fact, you can use `--incremental-lsn` to perform an incremental backup without even having the previous backup, if you know its *LSN*. Incremental backups simply read the pages and compare their *LSN* to the last backup's *LSN*. You still need a full backup to recover the incremental changes, however; without a full backup to act as a base, the incremental backups are useless.

Creating an Incremental Backup To make an incremental backup, begin with a full backup as usual. The **xtrabackup** binary writes a file called `xtrabackup_checkpoints` into the backup's target directory. This file contains a line showing the `to_lsn`, which is the database's *LSN* at the end of the backup. *Create the full backup* with a command such as the following:

```
xtrabackup --backup --target-dir=/data/backups/base --datadir=/var/lib/mysql/
```

If you want a usable full backup, use *innobackupex* since *xtrabackup* itself won't copy table definitions, triggers, or anything else that's not .ibd.

If you look at the `xtrabackup_checkpoints` file, you should see some contents similar to the following:

```
backup_type = full-backuped
from_lsn = 0
to_lsn = 1291135
```

Now that you have a full backup, you can make an incremental backup based on it. Use a command such as the following:

```
xtrabackup --backup --target-dir=/data/backups/inc1 \
--incremental-basedir=/data/backups/base --datadir=/var/lib/mysql/
```

The `/data/backups/inc1/` directory should now contain delta files, such as `ibdata1.delta` and `test/table1.ibd.delta`. These represent the changes since the `LSN 1291135`. If you examine the `xtrabackup_checkpoints` file in this directory, you should see something similar to the following:

```
backup_type = incremental
from_lsn = 1291135
to_lsn = 1291340
```

The meaning should be self-evident. It's now possible to use this directory as the base for yet another incremental backup:

```
xtrabackup --backup --target-dir=/data/backups/inc2 \
--incremental-basedir=/data/backups/inc1 --datadir=/var/lib/mysql/
```

Preparing the Incremental Backups The *--prepare* step for incremental backups is not the same as for normal backups. In normal backups, two types of operations are performed to make the database consistent: committed transactions are replayed from the log file against the data files, and uncommitted transactions are rolled back. You must skip the rollback of uncommitted transactions when preparing a backup, because transactions that were uncommitted at the time of your backup may be in progress, and it's likely that they will be committed in the next incremental backup. You should use the *--apply-log-only* option to prevent the rollback phase.

If you do not use the *--apply-log-only* option to prevent the rollback phase, then your incremental backups will be useless. After transactions have been rolled back, further incremental backups cannot be applied.

Beginning with the full backup you created, you can prepare it, and then apply the incremental differences to it. Recall that you have the following backups:

```
/data/backups/base
/data/backups/inc1
/data/backups/inc2
```

To prepare the base backup, you need to run *--prepare* as usual, but prevent the rollback phase:

```
xtrabackup --prepare --apply-log-only --target-dir=/data/backups/base
```

The output should end with some text such as the following:

```
101107 20:49:43  InnoDB: Shutdown completed; log sequence number 1291135
```

The log sequence number should match the `to_lsn` of the base backup, which you saw previously.

This backup is actually safe to *restore* as-is now, even though the rollback phase has been skipped. If you restore it and start *MySQL*, *InnoDB* will detect that the rollback phase was not performed, and it will do that in the background, as it usually does for a crash recovery upon start. It will notify you that the database was not shut down normally.

To apply the first incremental backup to the full backup, you should use the following command:

```
xtrabackup --prepare --apply-log-only --target-dir=/data/backups/base \
--incremental-dir=/data/backups/inc1
```

This applies the delta files to the files in `/data/backups/base`, which rolls them forward in time to the time of the incremental backup. It then applies the redo log as usual to the result. The final data is in `/data/backups/base`, not in the incremental directory. You should see some output such as the following:

```
incremental backup from 1291135 is enabled.
xtrabackup: cd to /data/backups/base/
xtrabackup: This target seems to be already prepared.
xtrabackup: xtrabackup_logfile detected: size=2097152, start_lsn=(1291340)
Applying /data/backups/inc1/ibdata1.delta ...
Applying /data/backups/inc1/test/table1.ibd.delta ...
.... snip
101107 20:56:30  InnoDB: Shutdown completed; log sequence number 1291340
```

Again, the *LSN* should match what you saw from your earlier inspection of the first incremental backup. If you restore the files from `/data/backups/base`, you should see the state of the database as of the first incremental backup.

Preparing the second incremental backup is a similar process: apply the deltas to the (modified) base backup, and you will roll its data forward in time to the point of the second incremental backup:

```
xtrabackup --prepare --target-dir=/data/backups/base \
--incremental-dir=/data/backups/inc2
```

Note: *--apply-log-only* should be used when merging all incrementals except the last one. That's why the previous line doesn't contain the *--apply-log-only* option. Even if the *--apply-log-only* was used on the last step, backup would still be consistent but in that case server would perform the rollback phase.

If you wish to avoid the notice that *InnoDB* was not shut down normally, when you have applied the desired deltas to the base backup, you can run *--prepare* again without disabling the rollback phase.

Incremental Backups with Log Archiving for XtraDB

Percona Server 5.6.11-60.3 has introduced a new feature called Log Archiving for XtraDB. This feature makes copies of the old log files before they are overwritten, thus saving all the redo log for a write workload.

When log archiving is enabled, it duplicates all redo log writes in a separate set of files in addition to normal redo log writing, creating new files as necessary.

Archived log file name format is `ib_log_archive_<startlsn>`. The start LSN marks the log sequence number when the archive was started. An example of the archived log files should look like this:

```
ib_log_archive_00000000010145937920
ib_log_archive_00000000010196267520
```

In order to enable this feature, variable *innodb_log_archive* should be set to `ON`. Once the feature has been enabled, directory specified with *innodb_log_arch_dir* (*MySQL* datadir by default) will contain the archived log files.

Creating the Backup To make an incremental backup, begin with a full backup as usual. The **xtrabackup** binary writes a file called `xtrabackup_checkpoints` into the backup's target directory. This file contains a line showing the `to_lsn`, which is the database's *LSN* at the end of the backup. *Create the full backup* with a command such as the following:

```
xtrabackup_56 --backup --target-dir=/data/backup/ --datadir=/var/lib/mysql/
```

If you want a usable full backup, use *innobackupex* since *xtrabackup* itself won't copy table definitions, triggers, or anything else that's not .ibd.

If you look at the `xtrabackup_checkpoints` file, you should see some contents similar to the following:

```
backup_type = full-backuped
from_lsn = 0
to_lsn = 1546908388
last_lsn = 1574827339
compact = 0
```

Using the Log Archiving to prepare the backup In order to prepare the backup we need to specify the directory that contains the archived logs with the *xtrabackup --innodb-log-arch-dir* option:

```
xtrabackup_56 --prepare --target-dir=/data/backup/ --innodb-log-arch-dir=/data/archived-logs/
```

This command will use archived logs, replay them against the backup folder and bring them up to date with the latest archived log.

Output should look like this:

```
xtrabackup_56 version 2.1.5 for MySQL server 5.6.11 Linux (x86_64) (revision id: 680)
xtrabackup: cd to /tmp/backup-01/
xtrabackup: using the following InnoDB configuration for recovery:
xtrabackup:   innodb_data_home_dir = ./
xtrabackup:   innodb_data_file_path = ibdata1:10M:autoextend
xtrabackup:   innodb_log_group_home_dir = ./
xtrabackup:   innodb_log_files_in_group = 2
xtrabackup:   innodb_log_file_size = 50331648
InnoDB: Allocated tablespace 4, old maximum was 0
xtrabackup: using the following InnoDB configuration for recovery:
xtrabackup:   innodb_data_home_dir = ./
xtrabackup:   innodb_data_file_path = ibdata1:10M:autoextend
xtrabackup:   innodb_log_group_home_dir = ./
xtrabackup:   innodb_log_files_in_group = 2
xtrabackup:   innodb_log_file_size = 50331648
xtrabackup: Starting InnoDB instance for recovery.
xtrabackup: Using 104857600 bytes for buffer pool (set by --use-memory parameter)
InnoDB: The InnoDB memory heap is disabled
InnoDB: Mutexes and rw_locks use GCC atomic builtins
InnoDB: Compressed tables use zlib 1.2.3
InnoDB: Not using CPU crc32 instructions
InnoDB: Initializing buffer pool, size = 100.0M
InnoDB: Completed initialization of buffer pool
InnoDB: Setting log file ./ib_logfile101 size to 48 MB
InnoDB: Setting log file ./ib_logfile1 size to 48 MB
InnoDB: Renaming log file ./ib_logfile101 to ./ib_logfile0
InnoDB: New log files created, LSN=1627148
InnoDB:  Starting archive recovery from a backup...
InnoDB: Allocated tablespace 4, old maximum was 0
InnoDB: Opened archived log file /var/lib/mysql/ib_log_archive_00000000000000045568
InnoDB: Starting an apply batch of log records to the database...
InnoDB: Progress in percent: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 2

...

InnoDB: Apply batch completed
InnoDB: Starting an apply batch of log records to the database...
InnoDB: Progress in percent: 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38
InnoDB: Apply batch completed
InnoDB: 1 transaction(s) which must be rolled back or cleaned up
InnoDB: in total 5063 row operations to undo
InnoDB: Trx id counter is 4096

[notice (again)]
If you use binary log and don't use any hack of group commit,
the binary log position seems to be:

xtrabackup: starting shutdown with innodb_fast_shutdown = 1
InnoDB: Starting shutdown...
InnoDB: Shutdown completed; log sequence number 2013229561
```

After this is completed successfully backup can be restored.

You can check the `xtrabackup_checkpoints` file and see that the backup_type has changed:

```
backup_type = full-prepared
from_lsn = 0
to_lsn = 1546908388
last_lsn = 1574827339
```

```
compact = 0
```

Note: Archived logs can be applied to backup data several times, for example to decrease the backup size or time required for preparing the backup.

Additional option is available if you need to restore a backup to specific point in time. By adding the *xtrabackup --to-archived-lsn* option you can specify the LSN to which the backup will be prepared.

```
xtrabackup_56 --prepare --target-dir=/data/backup/ --innodb-log-arch-dir=/data/archived-logs/ --to-ar
```

This will prepare the backup up to the specified Log Sequence Number.

Partial Backups

xtrabackup supports taking partial backups when the *innodb_file_per_table* option is enabled. There are two ways to create partial backups: matching the tables' names with a regular expression or providing a list of them in a file.

Warning: If any of the matched or listed tables is deleted during the backup, **xtrabackup** will fail.

For the purposes of this manual page, we will assume that there is a database named `test` which contains tables named `t1` and `t2`.

Using the *--tables* Option The first method is with the *--tables* option. The option's value is a regular expression that is matched against the fully qualified tablename, including the database name, in the form `databasename.tablename`.

To back up only tables in the `test` database, you can use the following command:

```
$ xtrabackup --backup --datadir=/var/lib/mysql --target-dir=/data/backups/ \
--tables="^test[.].*"
```

To back up only the table `test.t1`, you can use the following command:

```
$ xtrabackup --backup --datadir=/var/lib/mysql --target-dir=/data/backups/ \
--tables="^test[.]t1"
```

Using the *--tables-file* Option The `--tables-file` option specifies a file that can contain multiple table names, one table name per line in the file. Only the tables named in the file will be backed up. Names are matched exactly, case-sensitive, with no pattern or regular expression matching. The table names must be fully qualified, in `databasename.tablename` format.

$ echo "mydatabase.mytable" > /tmp/tables.txt $ xtrabackup –backup –tables-file=/tmp/tables.txt

Preparing the Backup When you use the *--prepare* option on a partial backup, you will see warnings about tables that don't exist. That is because these tables exist in the data dictionary inside InnoDB, but the corresponding *.ibd* files don't exist. They were not copied into the backup directory. These tables will be removed from the data dictionary, and when you restore the backup and start InnoDB, they will no longer exist and will not cause any errors or warnings to be printed to the log file.

An example of the error message you will see during the prepare phase follows.

```
InnoDB: Reading tablespace information from the .ibd files...
101107 22:31:30  InnoDB: Error: table 'test1/t'
InnoDB: in InnoDB data dictionary has tablespace id 6,
InnoDB: but tablespace with that id or name does not exist. It will be removed from data dictionary.
```

Compact Backups

When doing the backup of *InnoDB* tables it's possible to omit the secondary index pages. This will make the backups more compact and this way they will take less space on disk. The downside of this is that the backup prepare process takes longer as those secondary indexes need to be recreated. Difference in backup size depends on the size of the secondary indexes.

For example full backup taken without and with the *--compact* option:

```
#backup size without --compact
2.0G  xb_backup

#backup size taken with --compact option
1.4G  xb_compact_backup
```

Note: Compact backups are not supported for system table space, so in order to work correctly *innodb-file-per-table* option should be enabled.

This feature was introduced in *Percona XtraBackup* 2.1.

Creating Compact Backups To make a compact backup innobackupex needs to be started with the *--compact* option:

```
$ xtrabackup --backup --compact --target-dir=/data/backups
```

This will create a compact backup in the `/data/backups`.

If you check at the `xtrabackup-checkpoints` file in the `target-dir` folder, you should see something like:

```
backup_type = full-backuped
from_lsn = 0
to_lsn = 2888984349
last_lsn = 2888984349
compact = 1
```

When *--compact* wasn't used `compact` value will be `0`. This way it's easy to check if the backup contains the secondary index pages or not.

Preparing Compact Backups Preparing the compact require rebuilding the indexes as well. In order to prepare the backup a new option *--rebuild-indexes* should be used with *--apply-logs*:

```
$ xtrabackup --prepare --rebuild-indexes /data/backups/
```

Output, beside the standard **innobackupex** output, should contain the information about indexes being rebuilt, like:

```
[01] Checking if there are indexes to rebuild in table sakila/city (space id: 9)
[01]   Found index idx_fk_country_id
[01]   Rebuilding 1 index(es).
[01] Checking if there are indexes to rebuild in table sakila/country (space id: 10)
[01] Checking if there are indexes to rebuild in table sakila/customer (space id: 11)
```

```
[01]   Found index idx_fk_store_id
[01]   Found index idx_fk_address_id
[01]   Found index idx_last_name
[01]   Rebuilding 3 index(es).
```

Additionally, you can use the *--rebuild-threads* option to process tables in multiple threads when rebuilding indexes, e.g.:

```
$ xtrabackup --prepare --rebuild-indexes --rebuild-threads=16 /data/backups/
```

In this case *Percona XtraBackup* will create 16 worker threads with each thread rebuilding indexes for one table at a time. It will also show thread IDs for each message

```
Starting 16 threads to rebuild indexes.

[09] Checking if there are indexes to rebuild in table sakila/city (space id: 9)
[09]   Found index idx_fk_country_id
[10] Checking if there are indexes to rebuild in table sakila/country (space id: 10)
[11] Checking if there are indexes to rebuild in table sakila/customer (space id: 11)
[11]   Found index idx_fk_store_id
[11]   Found index idx_fk_address_id
[11]   Found index idx_last_name
[11]   Rebuilding 3 index(es).
```

Since *Percona XtraBackup* has no information when applying an incremental backup to a compact full one, on whether there will be more incremental backups applied to it later or not, rebuilding indexes needs to be explicitly requested by a user whenever a full backup with some incremental backups merged is ready to be restored. Rebuilding indexes unconditionally on every incremental backup merge is not an option, since it is an expensive operation.

Restoring Compact Backups The **xtrabackup** binary does not have any functionality for restoring a backup. That is up to the user to do. You might use **rsync** or **cp** to restore the files. You should check that the restored files have the correct ownership and permissions.

Other Reading

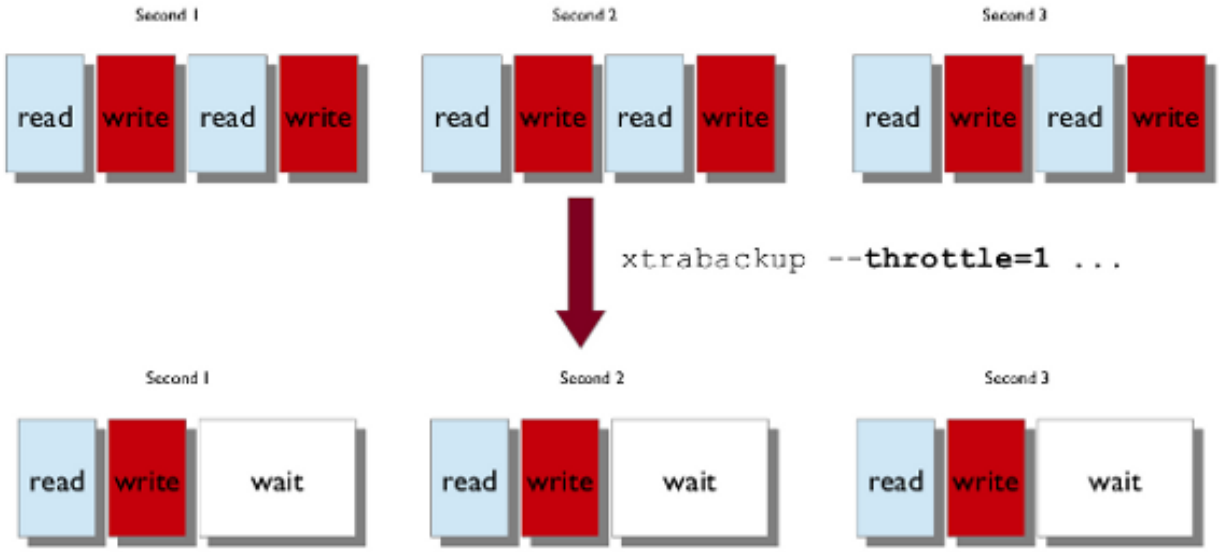
- Feature preview: Compact backups in Percona XtraBackup

Advanced Features

Throttling Backups

Although xtrabackup does not block your database's operation, any backup can add load to the system being backed up. On systems that do not have much spare I/O capacity, it might be helpful to throttle the rate at which xtrabackup reads and writes data. You can do this with the *--throttle* option, this option limits the number of I/O operations per second in 1 MB units.

Image below shows how throttling works when *--throttle* =1.



In `--backup` mode, this option limits the number of pairs of read-and-write operations per second that xtrabackup will perform. If you are creating an incremental backup, then the limit is the number of read IO operations per second.

By default, there is no throttling, and xtrabackup reads and writes data as quickly as it can. If you set too strict of a limit on the I/O operations, the backup might be so slow that it will never catch up with the transaction logs that InnoDB is writing, so the backup might never complete.

Scripting Backups With xtrabackup

The **xtrabackup** tool has several features to enable scripts to control it while they perform related tasks. The *innobackupex script* is one example, but **xtrabackup** is easy to control with your own command-line scripts too.

Suspending After Copying In backup mode, **xtrabackup** normally copies the log files in a background thread, copies the data files in a foreground thread, and then stops the log copying thread and finishes. If you use the *`--suspend-at-end`* option, instead of stopping the log thread and finishing, xtrabackup continues to copy the log files, and creates a file in the target directory called `xtrabackup_suspended`. As long as that file exists, xtrabackup will continue to watch the log files and copy them into the `xtrabackup_logfile` in the target directory. When the file is removed, **xtrabackup** will finish copying the log file and exit.

This functionality is useful for coordinating the InnoDB data backups with other actions. Perhaps the most obvious is copying the table definitions (the `.frm` files) so that the backup can be restored. You can start **xtrabackup** in the background, wait for the `xtrabackup_suspended` file to be created, and then copy any other files you need to complete the backup. This is exactly what the *innobackupex* tool does (it also copies MyISAM data and other files).

Generating Meta-Data It is a good idea for the backup to include all the information you need to restore the backup. The **xtrabackup** tool can print out the contents of a `my.cnf` file that are needed to restore the data and log files. If you add the *`--print-param`* option, it will print out something like the following:

```
# This MySQL options file was generated by XtraBackup.
[mysqld]
datadir = /data/mysql/
innodb_data_home_dir = /data/innodb/
innodb_data_file_path = ibdata1:10M:autoextend
innodb_log_group_home_dir = /data/innodb-logs/
```

You can redirect this output into a file in the target directory of the backup.

Agreeing on the Source Directory It's possible that the presence of a defaults file or other factors could cause **xtrabackup** to back up data from a different location than you expected. To prevent this, you can use *--print-param* to ask it where it will be copying data from. You can use the output to ensure that **xtrabackup** and your script are working on the same dataset.

Log Streaming You can instruct **xtrabackup** to omit copying data files, and simply stream the log file to its standard output instead with *--log-stream*. This automatically adds the *--suspend-at-end* option. Your script can then perform tasks such as streaming remote backups by piping the log files into an SSH connection and copying the data files to another server with a tool such as **rsync** or the *xbstream binary*.

Analyzing Table Statistics

The **xtrabackup** binary can analyze InnoDB data files in read-only mode to give statistics about them. To do this, you should use the *--stats* option. You can combine this with the *--tables* option to limit the files to examine. It also uses the *--use-memory* option.

You can perform the analysis on a running server, with some chance of errors due to the data being changed during analysis. Or, you can analyze a backup copy of the database. Either way, to use the statistics feature, you need a clean copy of the database including correctly sized log files, so you need to execute with *--prepare* twice to use this functionality on a backup.

The result of running on a backup might look like the following:

```
<INDEX STATISTICS>
  table: test/table1, index: PRIMARY, space id: 12, root page 3
  estimated statistics in dictionary:
    key vals: 25265338, leaf pages 497839, size pages 498304
  real statistics:
     level 2 pages: pages=1, data=5395 bytes, data/pages=32%
     level 1 pages: pages=415, data=6471907 bytes, data/pages=95%
        leaf pages: recs=25958413, pages=497839, data=7492026403 bytes, data/pages=91%
```

This can be interpreted as follows:

- The first line simply shows the table and index name and its internal identifiers. If you see an index named `GEN_CLUST_INDEX`, that is the table's clustered index, automatically created because you did not explicitly create a `PRIMARY KEY`.
- The estimated statistics in dictionary information is similar to the data that's gathered through `ANALYZE TABLE` inside of *InnoDB* to be stored as estimated cardinality statistics and passed to the query optimizer.
- The real statistics information is the result of scanning the data pages and computing exact information about the index.
- `The level <X> pages`: output means that the line shows information about pages at that level in the index tree. The larger `<X>` is, the farther it is from the leaf pages, which are level 0. The first line is the root page.
- The `leaf pages` output shows the leaf pages, of course. This is where the table's data is stored.
- The `external pages`: output (not shown) shows large external pages that hold values too long to fit in the row itself, such as long `BLOB` and `TEXT` values.
- The `recs` is the real number of records (rows) in leaf pages.
- The `pages` is the page count.

- The `data` is the total size of the data in the pages, in bytes.
- The `data/pages` is calculated as (`data` / (`pages` * `PAGE_SIZE`)) * 100%. It will never reach 100% because of space reserved for page headers and footers.

A more detailed example is posted as a MySQL Performance Blog post.

Script to Format Output The following script can be used to summarize and tabulate the output of the statistics information:

```
tabulate-xtrabackup-stats.pl

#!/usr/bin/env perl
use strict;
use warnings FATAL => 'all';
my $script_version = "0.1";

my $PG_SIZE = 16_384; # InnoDB defaults to 16k pages, change if needed.
my ($cur_idx, $cur_tbl);
my (%idx_stats, %tbl_stats);
my ($max_tbl_len, $max_idx_len) = (0, 0);
while ( my $line = <> ) {
   if ( my ($t, $i) = $line =~ m/table: (.*), index: (.*), space id:/ ) {
      $t =~ s!/!.!;
      $cur_tbl = $t;
      $cur_idx = $i;
      if ( length($i) > $max_idx_len ) {
         $max_idx_len = length($i);
      }
      if ( length($t) > $max_tbl_len ) {
         $max_tbl_len = length($t);
      }
   }
   elsif ( my ($kv, $lp, $sp) = $line =~ m/key vals: (\d+), \D*(\d+), \D*(\d+)/ ) {
      @{$idx_stats{$cur_tbl}->{$cur_idx}}{qw(est_kv est_lp est_sp)} = ($kv, $lp, $sp);
      $tbl_stats{$cur_tbl}->{est_kv} += $kv;
      $tbl_stats{$cur_tbl}->{est_lp} += $lp;
      $tbl_stats{$cur_tbl}->{est_sp} += $sp;
   }
   elsif ( my ($l, $pages, $bytes) = $line =~ m/(?:level (\d+)|leaf) pages:.*pages=(\d+), data=(\d+)
      $l ||= 0;
      $idx_stats{$cur_tbl}->{$cur_idx}->{real_pages} += $pages;
      $idx_stats{$cur_tbl}->{$cur_idx}->{real_bytes} += $bytes;
      $tbl_stats{$cur_tbl}->{real_pages} += $pages;
      $tbl_stats{$cur_tbl}->{real_bytes} += $bytes;
   }
}

my $hdr_fmt = "%${max_tbl_len}s %${max_idx_len}s %9s %10s %10s\n";
my @headers = qw(TABLE INDEX TOT_PAGES FREE_PAGES PCT_FULL);
printf $hdr_fmt, @headers;

my $row_fmt = "%${max_tbl_len}s %${max_idx_len}s %9d %10d %9.1f%%\n";
foreach my $t ( sort keys %tbl_stats ) {
   my $tbl = $tbl_stats{$t};
   printf $row_fmt, $t, "", $tbl->{est_sp}, $tbl->{est_sp} - $tbl->{real_pages},
      $tbl->{real_bytes} / ($tbl->{real_pages} * $PG_SIZE) * 100;
   foreach my $i ( sort keys %{$idx_stats{$t}} ) {
      my $idx = $idx_stats{$t}->{$i};
```

```
      printf $row_fmt, $t, $i, $idx->{est_sp}, $idx->{est_sp} - $idx->{real_pages},
         $idx->{real_bytes} / ($idx->{real_pages} * $PG_SIZE) * 100;
   }
}
```

Sample Script Output The output of the above Perl script, when run against the sample shown in the previously mentioned blog post, will appear as follows:

```
          TABLE           INDEX TOT_PAGES FREE_PAGES   PCT_FULL
art.link_out104                    832383      38561      86.8%
art.link_out104         PRIMARY    498304         49      91.9%
art.link_out104       domain_id     49600       6230      76.9%
art.link_out104     domain_id_2     26495       3339      89.1%
art.link_out104 from_message_id     28160        142      96.3%
art.link_out104    from_site_id     38848       4874      79.4%
art.link_out104   revert_domain    153984      19276      71.4%
art.link_out104    site_message     36992       4651      83.4%
```

The columns are the table and index, followed by the total number of pages in that index, the number of pages not actually occupied by data, and the number of bytes of real data as a percentage of the total size of the pages of real data. The first line in the above output, in which the `INDEX` column is empty, is a summary of the entire table.

Working with Binary Logs

The `xtrabackup` binary integrates with information that *InnoDB* stores in its transaction log about the corresponding binary log position for committed transactions. This enables it to print out the binary log position to which a backup corresponds, so you can use it to set up new replication slaves or perform point-in-time recovery.

Finding the Binary Log Position You can find the binary log position corresponding to a backup performing the `--prepare` process. If your backup is from a server with binary logging enabled, `xtrabackup` will create a file named `xtrabackup_binlog_info` in the target directory. This file contains the binary log file name and position of the exact point in the binary log to which the prepared backup corresponds.

You will also see output similar to the following during the prepare stage:

```
InnoDB: Last MySQL binlog file position 0 3252710, file name ./mysql-bin.000001
... snip ...
[notice (again)]
  If you use binary log and don't use any hack of group commit,
  the binary log position seems to be:
InnoDB: Last MySQL binlog file position 0 3252710, file name ./mysql-bin.000001
```

This output can also be found in the `xtrabackup_binlog_pos_innodb` file, but **it is only correct** when no other than *XtraDB* or *InnoDB* are used as storage engines.

If other storage engines are used (i.e. *MyISAM*), you should use the `xtrabackup_binlog_info` file to retrieve the position.

The message about hacking group commit refers to an early implementation of emulated group commit in *Percona Server*.

Point-In-Time Recovery To perform a point-in-time recovery from an `xtrabackup` backup, you should prepare and restore the backup, and then replay binary logs from the point shown in the `xtrabackup_binlog_info` file.

A more detailed procedure is found *here* (with **innobackupex**).

Setting Up a New Replication Slave To set up a new replica, you should prepare the backup, and restore it to the data directory of your new replication slave. Then in your `CHANGE MASTER TO` command, use the binary log filename and position shown in the `xtrabackup_binlog_info` file to start replication.

A more detailed procedure is found in *How to setup a slave for replication in 6 simple steps with Percona XtraBackup*.

Restoring Individual Tables

In server versions prior to 5.6, it is not possible to copy tables between servers by copying the files, even with *innodb_file_per_table*. However, with *Percona XtraBackup*, you can export individual tables from any *InnoDB* database, and import them into *Percona Server* with *XtraDB* or *MySQL* 5.6. (The source doesn't have to be *XtraDB* or or *MySQL* 5.6, but the destination does.) This only works on individual *.ibd* files, and cannot export a table that is not contained in its own *.ibd* file.

Let's see how to export and import the following table:

```
CREATE TABLE export_test (
  a int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Note: If you're running *Percona Server* version older than 5.5.10-20.1, variable innodb_expand_import should be used instead of innodb_import_table_from_xtrabackup.

Exporting the Table This table should have been created in *innodb_file_per_table* mode, so after taking a backup as usual with *--backup*, the *.ibd* file should exist in the target directory:

```
$ find /data/backups/mysql/ -name export_test.*
/data/backups/mysql/test/export_test.ibd
```

when you prepare the backup, add the extra parameter *--export* to the command. Here is an example:

```
$ xtrabackup --prepare --export --target-dir=/data/backups/mysql/
```

Now you should see a *.exp* file in the target directory:

```
$ find /data/backups/mysql/ -name export_test.*
/data/backups/mysql/test/export_test.exp
/data/backups/mysql/test/export_test.ibd
/data/backups/mysql/test/export_test.cfg
```

These three files are all you need to import the table into a server running *Percona Server* with *XtraDB* or *MySQL* 5.6.

Note: *MySQL* uses `.cfg` file which contains *InnoDB* dictionary dump in special format. This format is different from the `.exp` one which is used in *XtraDB* for the same purpose. Strictly speaking, a `.cfg` file is not required to import a tablespace to *MySQL* 5.6 or *Percona Server* 5.6. A tablespace will be imported successfully even if it is from another server, but *InnoDB* will do schema validation if the corresponding `.cfg` file is present in the same directory.

Importing the Table On the destination server running *Percona Server* with *XtraDB* and innodb_import_table_from_xtrabackup option enabled, or *MySQL* 5.6, create a table with the same structure, and then perform the following steps:

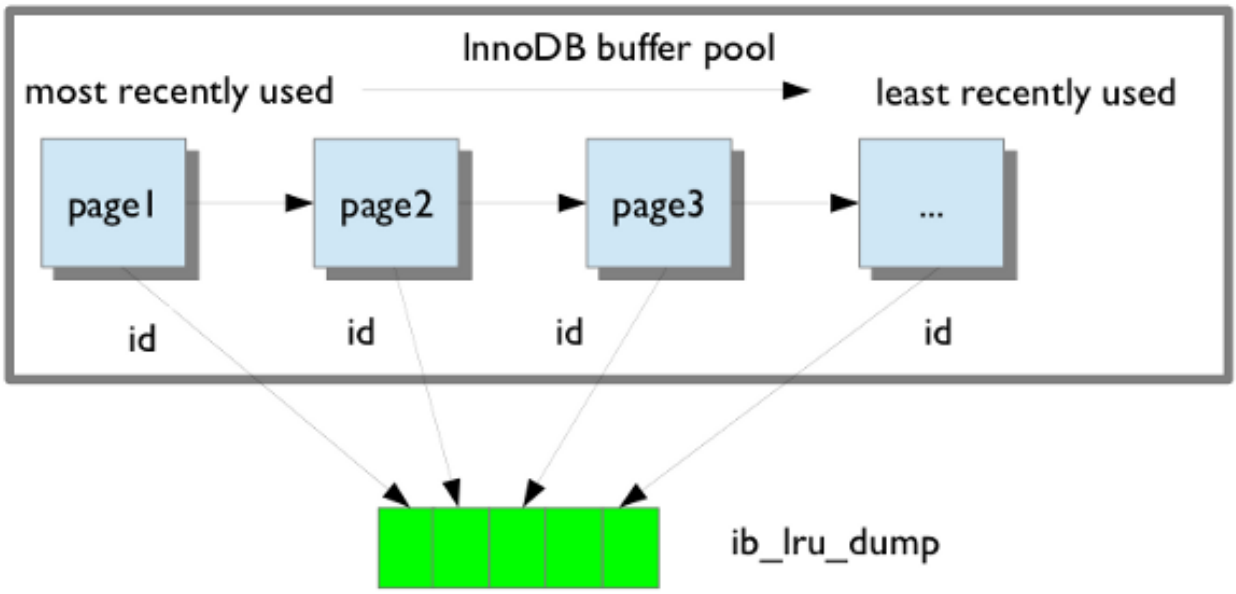
- Execute `ALTER TABLE test.export_test DISCARD TABLESPACE;`
  - If you see the following message, then you must enable *innodb_file_per_table* and create the table again: `ERROR 1030 (HY000): Got error -1 from storage engine`

- Copy the exported files to the `test/` subdirectory of the destination server's data directory
- Execute `ALTER TABLE test.export_test IMPORT TABLESPACE;`

The table should now be imported, and you should be able to `SELECT` from it and see the imported data.

LRU dump backup

This feature reduces the warm up time by restoring buffer pool state from `ib_lru_dump` file after restart. *Percona XtraBackup* discovers `ib_lru_dump` and backs it up automatically.



If the buffer restore option is enabled in `my.cnf` buffer pool will be in the warm state after backup is restored. To enable this set the variable innodb_buffer_pool_restore_at_startup =1 in Percona Server 5.5 or innodb_auto_lru_dump =1 in Percona Server 5.1.

Implementation

Limitations of xtrabackup

The `xtrabackup` binary has some limitations you should be aware of to ensure that your backups go smoothly and are recoverable.

- If the `xtrabackup_logfile` is larger than 4GB, the `--prepare` step will fail on 32-bit versions of `xtrabackup`.
- `xtrabackup` does not currently create new InnoDB log files (`ib_logfile0`, etc) during the initial `--prepare` step. You must prepare the backup a second time to do this, if you wish.
- `xtrabackup` copies only the InnoDB data and logs. It does not copy table definition files (`.frm` files), MyISAM data, users, privileges, or any other portions of the overall database that lie outside of the InnoDB data. To back up this data, you need a wrapper script such as *innobackupex*.
- `xtrabackup` doesn't understand the very old `--set-variable` `my.cnf` syntax that MySQL uses. See *Configuring xtrabackup*.

Implementation Details

This page contains notes on various internal aspects of the **xtrabackup** tool's operation.

File Permissions **xtrabackup** opens the source data files in read-write mode, although it does not modify the files. This means that you must run **xtrabackup** as a user who has permission to write the data files. The reason for opening the files in read-write mode is that **xtrabackup** uses the embedded *InnoDB* libraries to open and read the files, and *InnoDB* opens them in read-write mode because it normally assumes it is going to write to them.

Tuning the OS Buffers Because **xtrabackup** reads large amounts of data from the filesystem, it uses `posix_fadvise()` where possible, to instruct the operating system not to try to cache the blocks it reads from disk. Without this hint, the operating system would prefer to cache the blocks, assuming that `xtrabackup` is likely to need them again, which is not the case. Caching such large files can place pressure on the operating system's virtual memory and cause other processes, such as the database server, to be swapped out. The `xtrabackup` tool avoids this with the following hint on both the source and destination files:

```
posix_fadvise(file, 0, 0, POSIX_FADV_DONTNEED)
```

In addition, xtrabackup asks the operating system to perform more aggressive read-ahead optimizations on the source files:

```
posix_fadvise(file, 0, 0, POSIX_FADV_SEQUENTIAL)
```

Copying Data Files When copying the data files to the target directory, **xtrabackup** reads and writes 1MB of data at a time. This is not configurable. When copying the log file, **xtrabackup** reads and writes 512 bytes at a time. This is also not possible to configure, and matches InnoDB's behavior (workaround exists in *Percona Server* because it has an option to tune *`innodb_log_block_size`* for *XtraDB*, and in that case *Percona XtraBackup* will match the tuning).

After reading from the files, `xtrabackup` iterates over the 1MB buffer a page at a time, and checks for page corruption on each page with InnoDB's `buf_page_is_corrupted()` function. If the page is corrupt, it re-reads and retries up to 10 times for each page. It skips this check on the doublewrite buffer.

xtrabackup Exit Codes

The **xtrabackup** binary exits with the traditional success value of 0 after a backup when no error occurs. If an error occurs during the backup, the exit value is 1.

In certain cases, the exit value can be something other than 0 or 1, due to the command-line option code included from the *MySQL* libraries. An unknown command-line option, for example, will cause an exit code of 255.

References

The xtrabackup Option Reference

This page documents all of the command-line options for the **xtrabackup** binary.

Options

`--apply-log-only`
: This option causes only the redo stage to be performed when preparing a backup. It is very important for incremental backups.

--backup
Make a backup and place it in `--target-dir`. See *Creating a backup*.

--compact
Create a compact backup by skipping secondary index pages.

--compress
This option tells **xtrabackup** to compress all output data, including the transaction log file and meta data files, using the specified compression algorithm. The only currently supported algorithm is 'quicklz'. The resulting files have the qpress archive format, i.e. every **.qp* file produced by xtrabackup is essentially a one-file qpress archive and can be extracted and uncompressed by the qpress file archiver.

--compress-chunk-size=#
Size of working buffer(s) for compression threads in bytes. The default value is 64K.

--compress-threads=#
This option specifies the number of worker threads used by **xtrabackup** for parallel data compression. This option defaults to 1. Parallel compression ('–compress-threads') can be used together with parallel file copying ('–parallel'). For example, '–parallel=4 –compress –compress-threads=2' will create 4 IO threads that will read the data and pipe it to 2 compression threads.

--create-ib-logfile
This option is not currently implemented. To create the InnoDB log files, you must prepare the backup twice at present.

--datadir=DIRECTORY
The source directory for the backup. This should be the same as the datadir for your MySQL server, so it should be read from `my.cnf` if that exists; otherwise you must specify it on the command line.

--defaults-extra-file=[MY.CNF]
Read this file after the global files are read. Must be given as the first option on the command-line.

--defaults-file=[MY.CNF]
Only read default options from the given file. Must be given as the first option on the command-line. Must be a real file; it cannot be a symbolic link.

--defaults-group=GROUP-NAME
This option is to set the group which should be read from the configuration file. This is used by innobackupex if you use the *–defaults-group* option. It is needed for mysqld_multi deployments.

--export
Create files necessary for exporting tables. See *Restoring Individual Tables*.

--extra-lsndir=DIRECTORY
(for –backup): save an extra copy of the xtrabackup_checkpoints file in this directory.

--incremental-basedir=DIRECTORY
When creating an incremental backup, this is the directory containing the full backup that is the base dataset for the incremental backups.

--incremental-dir=DIRECTORY
When preparing an incremental backup, this is the directory where the incremental backup is combined with the full backup to make a new full backup.

--incremental-force-scan
When creating an incremental backup, force a full scan of the data pages in the instance being backuped even if the complete changed page bitmap data is available.

--incremental-lsn=LSN
When creating an incremental backup, you can specify the log sequence number (*LSN*) instead of specifying

--incremental-basedir. For databases created by *MySQL* and *Percona Server* 5.0-series versions, specify the *LSN* as two 32-bit integers in high:low format. For databases created in 5.1 and later, specify the *LSN* as a single 64-bit integer. ##ATTENTION##: If a wrong LSN value is specified (a user error which XtraBackup is unable to detect), the backup will be unusable. Be careful!

--innodb-log-arch-dir=DIRECTORY
This option is used to specify the directory containing the archived logs. It can only be used with the *xtrabackup --prepare* option.

--innodb-miscellaneous
There is a large group of InnoDB options that are normally read from the *my.cnf* configuration file, so that xtrabackup boots up its embedded InnoDB in the same configuration as your current server. You normally do not need to specify these explicitly. These options have the same behavior that they have in InnoDB or XtraDB. They are as follows:

```
--innodb-adaptive-hash-index
--innodb-additional-mem-pool-size
--innodb-autoextend-increment
--innodb-buffer-pool-size
--innodb-checksums
--innodb-data-file-path
--innodb-data-home-dir
--innodb-doublewrite-file
--innodb-doublewrite
--innodb-extra-undoslots
--innodb-fast-checksum
--innodb-file-io-threads
--innodb-file-per-table
--innodb-flush-log-at-trx-commit
--innodb-flush-method
--innodb-force-recovery
--innodb-io-capacity
--innodb-lock-wait-timeout
--innodb-log-buffer-size
--innodb-log-files-in-group
--innodb-log-file-size
--innodb-log-group-home-dir
--innodb-max-dirty-pages-pct
--innodb-open-files
--innodb-page-size
--innodb-read-io-threads
--innodb-write-io-threads
```

--log-copy-interval=#
This option specifies time interval between checks done by log copying thread in milliseconds (default is 1 second).

--log-stream
Makes xtrabackup not copy data files, and output the contents of the InnoDB log files to STDOUT until the *--suspend-at-end* file is deleted. This option enables *--suspend-at-end* automatically.

--no-defaults
Don't read default options from any option file. Must be given as the first option on the command-line.

--parallel=#
This option specifies the number of threads to use to copy multiple data files concurrently when creating a backup. The default value is 1 (i.e., no concurrent transfer).

--prepare
Makes **xtrabackup** perform recovery on a backup created with *--backup*, so that it is ready to use. See

preparing a backup.

--print-defaults
Print the program argument list and exit. Must be given as the first option on the command-line.

--print-param
Makes **xtrabackup** print out parameters that can be used for copying the data files back to their original locations to restore them. See *Scripting Backups With xtrabackup*.

--rebuild_indexes
Rebuild secondary indexes in InnoDB tables after applying the log. Only has effect with –prepare.

--rebuild_threads=#
Use this number of threads to rebuild indexes in a compact backup. Only has effect with –prepare and –rebuild-indexes.

--stats
Causes **xtrabackup** to scan the specified data files and print out index statistics.

--stream=name
Stream all backup files to the standard output in the specified format. Currently supported formats are 'xbstream' and 'tar'.

--suspend-at-end
Causes **xtrabackup** to create a file called `xtrabackup_suspended` in the *--target-dir*. Instead of exiting after copying data files, **xtrabackup** continues to copy the log file, and waits until the `xtrabackup_suspended` file is deleted. This enables xtrabackup and other programs to coordinate their work. See *Scripting Backups With xtrabackup*.

--tables=name
A regular expression against which the full tablename, in `databasename.tablename` format, is matched. If the name matches, the table is backed up. See *partial backups*.

--tables-file=name
A file containing one table name per line, in databasename.tablename format. The backup will be limited to the specified tables. See *Scripting Backups With xtrabackup*.

--target-dir=DIRECTORY
This option specifies the destination directory for the backup. If the directory does not exist, **xtrabackup** creates it. If the directory does exist and is empty, **xtrabackup** will succeed. **xtrabackup** will not overwrite existing files, however; it will fail with operating system error 17, `file exists`.

If this option is a relative path, it is interpreted as being relative to the current working directory from which **xtrabackup** is executed.

--throttle=#
This option limits *--backup* to the specified number of read+write pairs of operations per second. See *throttling a backup*.

--tmpdir=name
This option is currently not used for anything except printing out the correct tmpdir parameter when *--print-param* is used.

--to-archived-lsn=LSN
This option is used to specify the LSN to which the logs should be applied when backups are being prepared. It can only be used with the *xtrabackup --prepare* option.

--use-memory=#
This option affects how much memory is allocated for preparing a backup with *--prepare*, or analyzing statistics with *--stats*. Its purpose is similar to *innodb_buffer_pool_size*. It does not do the same thing as the similarly named option in Oracle's InnoDB Hot Backup tool. The default value is 100MB, and if you have

enough available memory, 1GB to 2GB is a good recommended value. Multiples are supported providing the unit (e.g. 1MB, 1M, 1GB, 1G).

--version
This option prints **xtrabackup** version and exits.

3.1.3 The xbstream Binary

To support simultaneous compression and streaming, a new custom streaming format called xbstream was introduced to *Percona XtraBackup* in addition to the TAR format. That was required to overcome some limitations of traditional archive formats such as tar, cpio and others which did not allow streaming dynamically generated files, for example dynamically compressed files. Other advantages of xbstream over traditional streaming/archive format include ability to stream multiple files concurrently (so it is possible to use streaming in the xbstream format together with the –parallel option) and more compact data storage.

This utility has a tar-like interface:

- with the '-x' option it extracts files from the stream read from its standard input to the current directory unless specified otherwise with the '-C' option.
- with the '-c' option it streams files specified on the command line to its standard output.

The utility also tries to minimize its impact on the OS page cache by using the appropriate posix_fadvise() calls when available.

When compression is enabled with **xtrabackup** all data is being compressed, including the transaction log file and meta data files, using the specified compression algorithm. The only currently supported algorithm is 'quicklz'. The resulting files have the qpress archive format, i.e. every *.qp file produced by xtrabackup is essentially a one-file qpress archive and can be extracted and uncompressed by the qpress file archiver. This means that there is no need to uncompress entire backup to restore a single table as with tar.gz.

Files can be decompressed using the **qpress** tool that can be downloaded from here. Qpress supports multi-threaded decompression.

3.1.4 The xbcrypt binary

To support encryption and decryption of the backups, a new tool `xbcrypt` was introduced to *Percona XtraBackup*.

This utility has been modeled after *The xbstream Binary* to perform encryption and decryption outside of *Percona XtraBackup*. Xbcrypt has following command line options:

-d, --decrypt
Decrypt data input to output.

-i, --input=name
Optional input file. If not specified, input will be read from standard input.

-o, --output=name
Optional output file. If not specified, output will be written to standard output.

-a, --encrypt-algo=name
Encryption algorithm.

-k, --encrypt-key=name
Encryption key.

-f, --encrypt-key-file=name
File which contains encryption key.

-s, --encrypt-chunk-size=#
Size of working buffer for encryption in bytes. The default value is 64K.

-v, --verbose
Display verbose status output.

3.1.5 How *Percona XtraBackup* Works

Percona XtraBackup is based on *InnoDB*'s crash-recovery functionality. It copies your *InnoDB* data files, which results in data that is internally inconsistent; but then it performs crash recovery on the files to make them a consistent, usable database again.

This works because *InnoDB* maintains a redo log, also called the transaction log. This contains a record of every change to InnoDB's data. When *InnoDB* starts, it inspects the data files and the transaction log, and performs two steps. It applies committed transaction log entries to the data files, and it performs an undo operation on any transactions that modified data but did not commit.

Percona XtraBackup works by remembering the log sequence number (*LSN*) when it starts, and then copying away the data files. It takes some time to do this, so if the files are changing, then they reflect the state of the database at different points in time. At the same time, *Percona XtraBackup* runs a background process that watches the transaction log files, and copies changes from it. *Percona XtraBackup* needs to do this continually because the transaction logs are written in a round-robin fashion, and can be reused after a while. *Percona XtraBackup* needs the transaction log records for every change to the data files since it began execution.

The above is the backup process. Next is the prepare process. During this step, *Percona XtraBackup* performs crash recovery against the copied data files, using the copied transaction log file. After this is done, the database is ready to restore and use.

The above process is implemented in the **xtrabackup** compiled binary program. The **innobackupex** program adds more convenience and functionality by also permitting you to back up *MyISAM* tables and *.frm* files. It starts **xtrabackup**, waits until it finishes copying files, and then issues `FLUSH TABLES WITH READ LOCK` to prevent further changes to *MySQL*'s data and flush all *MyISAM* tables to disk. It holds this lock, copies the *MyISAM* files, and then releases the lock.

The backed-up *MyISAM* and *InnoDB* tables will eventually be consistent with each other, because after the prepare (recovery) process, *InnoDB*'s data is rolled forward to the point at which the backup completed, not rolled back to the point at which it started. This point in time matches where the `FLUSH TABLES WITH READ LOCK` was taken, so the *MyISAM* data and the prepared *InnoDB* data are in sync.

The **xtrabackup** and **innobackupex** tools both offer many features not mentioned in the preceding explanation. Each tool's functionality is explained in more detail on its manual page. In brief, though, the tools permit you to do operations such as streaming and incremental backups with various combinations of copying the data files, copying the log files, and applying the logs to the data.

Percona XtraBackup is a set of following tools:

innobackupex a wrapper script that provides functionality to backup a whole *MySQL* database instance with *MyISAM*, *InnoDB*, and *XtraDB* tables.

xtrabackup a compiled *C* binary, which copies only *InnoDB* and *XtraDB* data

xbcrypt utility used for encrypting and decrypting backup files.

xbstream utility that allows streaming and extracting files to/from the *xbstream* format.

It is possible to use the **xtrabackup** binary alone, however, the recommend way is using it through the **innobackupex** wrapper script and let it execute **xtrabackup** for you. It might be helpful to first learn *how to use innobackupex*, and then learn *how to use xtrabackup* for having a better low-level understanding or control of the tool if needed.

CHAPTER

FOUR

TUTORIALS, RECIPES, HOW-TOS

4.1 How-tos and Recipes

4.1.1 Recipes for innobackupex

Make a Local Full Backup (Create, Prepare and Restore)

Create the Backup

This is the simplest use case. It copies all your *MySQL* data into the specified directory. Here is how to make a backup of all the databases in the *datadir* specified in your *my.cnf*. It will put the backup in a time stamped subdirectory of `/data/backups/`, in this case, `/data/backups/2010-03-13_02-42-44`,

```
$ innobackupex /data/backups
```

There is a lot of output, but you need to make sure you see this at the end of the backup. If you don't see this output, then your backup failed:

```
100313 02:43:07  innobackupex: completed OK!
```

Prepare the Backup

To prepare the backup use the `--apply-log` option and specify the timestamped subdirectory of the backup. To speed up the apply-log process, we using the `--use-memory` option is recommended:

```
$ innobackupex --use-memory=4G --apply-log /data/backups/2010-03-13_02-42-44/
```

You should check for a confirmation message:

```
100313 02:51:02  innobackupex: completed OK!
```

Now the files in `/data/backups/2010-03-13_02-42-44` is ready to be used by the server.

Restore the Backup

To restore the already-prepared backup, first stop the server and then use the `--copy-back` function of **innobackupex**:

```
innobackupex --copy-back /data/backups/2010-03-13_02-42-44/
## Use chmod to correct the permissions, if necessary!
```

This will copy the prepared data back to its original location as defined by the `datadir` in your *my.cnf*.

After the confirmation message:

```
100313 02:58:44  innobackupex: completed OK!
```

you should check the file permissions after copying the data back. You may need to adjust them with something like:

```
$ chown -R mysql:mysql /var/lib/mysql
```

Now the *datadir* contains the restored data. You are ready to start the server.

Make a Streaming Backup

Stream mode sends the backup to `STDOUT` in tar format instead of copying it to the directory named by the first argument. You can pipe the output to **gzip**, or across the network to another server.

To extract the resulting tar file, you **must** use the `-i` option, such as `tar -ixvf backup.tar`.

Warning: Remember to use the `-i` option for extracting a tarred backup. For more information, see *Streaming and Compressing Backups*.

Here are some examples using `tar` option for streaming:

- Stream the backup into a tar archived named 'backup.tar'

  ```
  $ innobackupex --stream=tar ./ > backup.tar
  ```

- The same, but compress it

  ```
  $ innobackupex --stream=tar ./ | gzip - > backup.tar.gz
  ```

- Encrypt the backup

  ```
  $ innobackupex --stream=tar . | gzip - | openssl des3 -salt -k "password" > backup.tar.gz.des3
  ```

- Send it to another server instead of storing it locally

  ```
  $ innobackupex --stream=tar ./ | ssh user@desthost "cat - > /data/backups/backup.tar"
  ```

- The same thing with can be done with the ''netcat''.

  ```
  ## On the destination host:
  $ nc -l 9999 | cat - > /data/backups/backup.tar
  ## On the source host:
  $ innobackupex --stream=tar ./ | nc desthost 9999
  ```

- The same thing, but done as a one-liner:

  ```
  $ ssh user@desthost "( nc -l 9999 > /data/backups/backup.tar & )" \
  && innobackupex --stream=tar ./  |  nc desthost 9999
  ```

- Throttling the throughput to 10MB/sec. This requires the 'pv' tools; you can find them at the official site or install it from the distribution package ("apt-get install pv")

  ```
  $ innobackupex --stream=tar ./ | pv -q -L10m \
  | ssh user@desthost "cat - > /data/backups/backup.tar"
  ```

- Checksumming the backup during the streaming

```
## On the destination host:
$ nc -l 9999 | tee >(sha1sum > destination_checksum) > /data/backups/backup.tar
## On the source host:
$ innobackupex --stream=tar ./ | tee >(sha1sum > source_checksum) | nc desthost 9999
## compare the checksums
## On the source host:
$ cat source_checksum
65e4f916a49c1f216e0887ce54cf59bf3934dbad  -
## On the destination host:
$ destination_checksum
65e4f916a49c1f216e0887ce54cf59bf3934dbad  -
```

Examples using *xbstream* option for streaming:

- Stream the backup into a tar archived named 'backup.xbstream

  ```
  innobackupex --stream=xbstream ./ > backup.xbstream
  ```

- The same but with compression

  ```
  innobackupex --stream=xbstream --compress ./ > backup.xbstream
  ```

- To unpack the backup to the current directory:

  ```
  xbstream -x <  backup.xbstream
  ```

- Sending backup compressed directly to another host and unpacking it:

  ```
  innobackupex --compress --stream=xbstream ./ | ssh user@otherhost "xbstream -x"
  ```

- Parallel compression with parallel copying backup

  ```
  innobackupex --compress --compress-threads=8 --stream=xbstream --parallel=4 ./ > backup.xbstream
  ```

Making an Incremental Backup

Every incremental backup starts with a full one, which we will call the *base backup*:

```
innobackupex --user=USER --password=PASSWORD /path/to/backup/dir/
```

Note that the full backup will be in a timestamped subdirectory of `/path/to/backup/dir/` (e.g. `/path/to/backup/dir/2011-12-24_23-01-00/`).

Assuming that variable `$FULLBACKUP` contains `/path/to/backup/dir/2011-5-23_23-01-18`, let's do an incremental backup an hour later:

```
innobackupex --incremental /path/to/inc/dir \
  --incremental-basedir=$FULLBACKUP --user=USER --password=PASSWORD
```

Now, the incremental backup should be in `/path/to/inc/dir/2011-12-25_00-01-00/`. Let's call `$INCREMENTALBACKUP=2011-5-23_23-50-10`.

Preparing incremental backups is a bit different than full ones:

First you have to replay the committed transactions on each backup,

```
innobackupex --apply-log --redo-only $FULLBACKUP \
 --use-memory=1G --user=USER --password=PASSWORD
```

The *--use-memory* option is not necessary, it will speed up the process if it is used (provided that the amount of RAM given is available).

If everything went fine, you should see an output similar to:

```
111225 01:10:12 InnoDB: Shutdown completed; log sequence number 91514213
```

Now apply the incremental backup to the base backup, by issuing:

```
innobackupex --apply-log --redo-only $FULLBACKUP
 --incremental-dir=$INCREMENTALBACKUP
 --use-memory=1G --user=DVADER --password=D4RKS1D3
```

Note the `$INCREMENTALBACKUP`.

The final data will be in the base backup directory, not in the incremental one. In this example, `/path/to/backup/dir/2011-12-24_23-01-00` or `$FULLBACKUP`.

If you want to apply more incremental backups, repeat this step with the next one. It is important that you do this in the chronological order in which the backups were done.

You can check the file xtrabackup_checkpoints at the directory of each one.

They should look like: (in the base backup)

```
backup_type = full-backuped
from_lsn = 0
to_lsn = 1291135
```

and in the incremental ones:

```
backup_type = incremental
from_lsn = 1291135
to_lsn = 1291340
```

The `to_lsn` number must match the `from_lsn` of the next one.

Once you put all the parts together, you can prepare again the full backup (base + incrementals) once again to rollback the pending transactions:

```
innobackupex-1.5.1 --apply-log $FULLBACKUP --use-memory=1G \
  --user=$USERNAME --password=$PASSWORD
```

Now your backup is ready to be used immediately after restoring it. This preparation step is optional, as if you restore it without doing it, the database server will assume that a crash occurred and will begin to rollback the uncommitted transaction (causing some downtime which can be avoided).

Making a Compressed Backup

In order to make a compressed backup you'll need to use *--compress* option

```
$ innobackupex --compress /data/backup
```

If you want to speed up the compression you can use the parallel compression, which can be enabled with *--compress-threads=#* option. Following example will use four threads for compression:

```
$ innobackupex --compress --compress-threads=4 /data/backup
```

Output should look like this

```
...
[01] Compressing ./imdb/comp_cast_type.ibd to /data/backup/2013-08-01_11-24-04/./imdb/comp_cast_type
[01]        ...done
[01] Compressing ./imdb/aka_name.ibd to /data/backup/2013-08-01_11-24-04/./imdb/aka_name.ibd.qp
[01]        ...done
...
130801 11:50:24  innobackupex: completed OK
```

Preparing the backup

Before you can prepare the backup you'll need to uncompress all the files with qpress (which is available from Percona Software repositories). You can use following one-liner to uncompress all the files:

```
$ for bf in `find . -iname "*\.qp"`; do qpress -d $bf $(dirname $bf) && rm $bf; done
```

In *Percona XtraBackup* 2.1.4 new *innobackupex --decompress* option has been implemented that can be used to decompress the backup:

```
$ innobackupex --decompress /data/backup/2013-08-01_11-24-04/
```

This option will remove the original compressed files and leave the uncompressed ones in the same location.

Note: In order to successfully use the *innobackupex --decompress* option, qpress binary needs to installed and within the path. *innobackupex --parallel* can be used with *innobackupex --decompress* option to decompress multiple files simultaneously.

When the files are uncompressed you can prepare the backup with the *--apply-log* option:

```
$ innobackupex --apply-log /data/backup/2013-08-01_11-24-04/
```

You should check for a confirmation message:

```
130802 02:51:02  innobackupex: completed OK!
```

Now the files in `/data/backups/2013-08-01_11-24-04` is ready to be used by the server.

Restoring the backup

Once the backup has been prepared you can use the *--copy-back* to restore the backup.

```
$ innobackupex --copy-back /data/backups/2013-08-01_11-24-04/
```

This will copy the prepared data back to its original location as defined by the `datadir` in your *my.cnf*.

After the confirmation message:

```
130802 02:58:44  innobackupex: completed OK!
```

you should check the file permissions after copying the data back. You may need to adjust them with something like:

```
$ chown -R mysql:mysql /var/lib/mysql
```

Now the *datadir* contains the restored data. You are ready to start the server.

Backing Up and Restoring Individual Partitions

Percona XtraBackup features *partial backups*, which means that you may backup individual partitions as well because from the storage engines perspective partitions are regular tables with specially formatted names. The only requirement for this feature is having the *innodb_file_per_table* option enabled in the server.

There is only one caveat about using this kind of backup: you can't copy back the prepared backup. Restoring partial backups should be done by importing the tables, and not by using the traditional `--copy-back` option. Although there are some scenarios where restoring can be done by copying back the files, this may be lead to database inconsistencies in many cases and it is not the recommended way to do it.

Creating the backup

There are three ways of specifying which part of the whole data will be backed up: regular expressions (`--include`), enumerating the tables in a file (`--tables-file`) or providing a list of databases (`--databases`). In this example `--include` option will be used.

The regular expression provided to this option will be matched against the fully qualified tablename, including the database name, in the form `databasename.tablename`.

For example, this will back up the partition `p4` from the table `name` located in the database `imdb`:

```
$ innobackupex --include='^imdb[.]name#P#p4' /mnt/backup/
```

This will create a timestamped directory with the usual files that **innobackupex** creates, but only the data files related to the tables matched.

Output of the **innobackupex** will list the skipped tables

```
...
[01] Skipping ./imdb/person_info.ibd
[01] Skipping ./imdb/name#P#p5.ibd
[01] Skipping ./imdb/name#P#p6.ibd
...
imdb.person_info.frm is skipped because it does not match ^imdb[.]name#P#p4.
imdb.title.frm is skipped because it does not match ^imdb[.]name#P#p4.
imdb.company_type.frm is skipped because it does not match ^imdb[.]name#P#p4.
...
```

Note that this option is passed to *xtrabackup --tables* and is matched against each table of each database, the directories of each database will be created even if they are empty.

Preparing the backup

For preparing partial backups, the procedure is analogous to *restoring individual tables* : apply the logs and use the `--export` option:

```
$ innobackupex --apply-log --export /mnt/backup/2012-08-28_10-29-09
```

You may see warnings in the output about tables that don't exists. This is because *InnoDB*-based engines stores its data dictionary inside the tablespace files besides the *.frm* files. **innobackupex** will use **xtrabackup** to remove the missing tables (those that haven't been selected in the partial backup) from the data dictionary in order to avoid future warnings or errors:

```
InnoDB: in InnoDB data dictionary has tablespace id 51,
InnoDB: but tablespace with that id or name does not exist. It will be removed from data dictionary.
120828 10:25:28  InnoDB: Waiting for the background threads to start
```

```
120828 10:25:29 Percona XtraDB (http://www.percona.com) 1.1.8-20.1 started; log sequence number 10098
xtrabackup: export option is specified.
xtrabackup: export metadata of table 'imdb/name#P#p4' to file `./imdb/name#P#p4.exp` (1 indexes)
xtrabackup:     name=PRIMARY, id.low=73, page=3
```

You should also see the notification of the creation of a file needed for importing (*.exp* file) for each table included in the partial backup:

```
xtrabackup: export option is specified.
xtrabackup: export metadata of table 'imdb/name#P#p4' to file `./imdb/name#P#p4.exp` (1 indexes)
xtrabackup:     name=PRIMARY, id.low=73, page=3
```

Note that you can use the *--export* option with *--apply-log* to an already-prepared backup in order to create the *.exp* files.

Finally, check the for the confirmation message in the output:

```
120828 19:25:38  innobackupex: completed OK!
```

Restoring from the backups

Restoring should be done by *importing the tables* in the partial backup to the server.

Note: Improved table/partition import is only available in *Percona Server* and *MySQL* 5.6, this means that partitions which were backed up from different server can be imported as well. For versions older than *MySQL* 5.6 only partitions from that server can be imported with some important limitations. There should be no DROP/CREATE/TRUNCATE/ALTER TABLE commands issued between taking the backup and importing the partition.

First step is to create new table in which data will be restored

```
mysql> CREATE TABLE `name_p4` (
`id` int(11) NOT NULL AUTO_INCREMENT,
`name` text NOT NULL,
`imdb_index` varchar(12) DEFAULT NULL,
`imdb_id` int(11) DEFAULT NULL,
`name_pcode_cf` varchar(5) DEFAULT NULL,
`name_pcode_nf` varchar(5) DEFAULT NULL,
`surname_pcode` varchar(5) DEFAULT NULL,
PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=2812744 DEFAULT CHARSET=utf8
```

To restore the partition from the backup tablespace needs to be discarded for that table:

```
mysql>  ALTER TABLE name_p4 DISCARD TABLESPACE;
```

Next step is to copy the *.exp* and *ibd* files from the backup to *MySQL* data directory:

```
$ cp /mnt/backup/2012-08-28_10-29-09/imdb/name#P#p4.exp /var/lib/mysql/imdb/name_p4.exp
$ cp /mnt/backup/2012-08-28_10-29-09/imdb/name#P#p4.ibd /var/lib/mysql/imdb/name_p4.ibd
```

Note: Make sure that the copied files can be accessed by the user running the *MySQL*.

If you're running the *Percona Server* make sure that variable *innodb_import_table_from_xtrabackup* is enabled:

```
mysql> SET GLOBAL innodb_import_table_from_xtrabackup=1;
```

Last step is to import the tablespace:

```
mysql>  ALTER TABLE name_p4 IMPORT TABLESPACE;
```

Restoring from the backups in version 5.6 The problem with server versions up to 5.5 is that there is no server support to import either individual partitions or all partitions of a partitioned table, so partitions could only be imported as independent tables. In *MySQL* and *Percona Server* 5.6 it is possible to exchange individual partitions with independent tables through `ALTER TABLE ... EXCHANGE PARTITION` command.

Note: In *Percona Server* 5.6 variable *`innodb_import_table_from_xtrabackup`* has been removed in favor of *MySQL* Transportable Tablespaces implementation.

When importing an entire partitioned table, first import all (sub)partitions as independent tables:

```
mysql> CREATE TABLE `name_p4` (
`id` int(11) NOT NULL AUTO_INCREMENT,
`name` text NOT NULL,
`imdb_index` varchar(12) DEFAULT NULL,
`imdb_id` int(11) DEFAULT NULL,
`name_pcode_cf` varchar(5) DEFAULT NULL,
`name_pcode_nf` varchar(5) DEFAULT NULL,
`surname_pcode` varchar(5) DEFAULT NULL,
PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=2812744 DEFAULT CHARSET=utf8
```

To restore the partition from the backup tablespace needs to be discarded for that table:

```
mysql>  ALTER TABLE name_p4 DISCARD TABLESPACE;
```

Next step is to copy the `.cfg` and `.ibd` files from the backup to *MySQL* data directory:

```
$ cp /mnt/backup/2013-07-18_10-29-09/imdb/name#P#p4.cfg /var/lib/mysql/imdb/name_p4.cfg
$ cp /mnt/backup/2013-07-18_10-29-09/imdb/name#P#p4.ibd /var/lib/mysql/imdb/name_p4.ibd
```

Last step is to import the tablespace:

```
mysql>  ALTER TABLE name_p4 IMPORT TABLESPACE;
```

We can now create the empty partitioned table with exactly the same schema as the table being imported:

```
mysql> CREATE TABLE name2 LIKE name;
```

Then swap empty partitions from the newly created table with individual tables corresponding to partitions that have been exported/imported on the previous steps

```
mysql> ALTER TABLE name2 EXCHANGE PARTITION p4 WITH TABLE name_p4;
```

In order for this operation to be successful following conditions have to be met.

4.1.2 Recipes for xtrabackup

Making a Full Backup

Backup the InnoDB data and log files - located in `/var/lib/mysql/` - to `/data/backups/mysql/` (destination). Then, prepare the backup files to be ready to restore

or use (make the data files consistent).

Make a backup:

```
xtrabackup --backup --target-dir=/data/backups/mysql/
```

Prepare the backup twice:

```
xtrabackup --prepare --target-dir=/data/backups/mysql/
xtrabackup --prepare --target-dir=/data/backups/mysql/
```

Success Criterion

- The exit status of xtrabackup is 0.
- In the second `--prepare` step, you should see InnoDB print messages similar to `Log file ./ib_logfile0 did not exist: new to be created`, followed by a line indicating the log file was created (creating new logs is the purpose of the second preparation).

Notes

- You might want to set the `--use-memory` option to something similar to the size of your buffer pool, if you are on a dedicated server that has enough free memory. More details *here*.
- A more detailed explanation is *here*

Making an Incremental Backup

Backup all the InnoDB data and log files - located in `/var/lib/mysql/` - **once**, then make two daily incremental backups in `/data/backups/mysql/` (destination). Finally, prepare the backup files to be ready to restore or use.

Create one full backup

Making an incremental backup requires a full backup as a base:

```
xtrabackup --backup --target-dir=/data/backups/mysql/
```

It is important that you **do not run** the `--prepare` command yet.

Create two incremental backups

Suppose the full backup is on Monday, and you will create an incremental one on Tuesday:

```
xtrabackup --backup --target-dir=/data/backups/inc/tue/ \
      --incremental-basedir=/data/backups/mysql/
```

and the same policy is applied on Wednesday:

```
xtrabackup --backup --target-dir=/data/backups/inc/wed/ \
       --incremental-basedir=/data/backups/inc/tue/
```

Prepare the base backup

Prepare the base backup (Monday's backup):

```
xtrabackup --prepare --apply-log-only --target-dir=/data/backups/mysql/
```

Roll forward the base data to the first increment

Roll Monday's data forward to the state on Tuesday:

```
xtrabackup --prepare --apply-log-only --target-dir=/data/backups/mysql/ \
   --incremental-dir=/data/backups/inc/tue/
```

Roll forward again to the second increment

Roll forward again to the state on Wednesday:

```
xtrabackup --prepare --apply-log-only --target-dir=/data/backups/mysql/ \
   --incremental-dir=/data/backups/inc/wed/
```

Prepare the whole backup to be ready to use

Create the new logs by preparing it:

```
xtrabackup --prepare --target-dir=/data/backups/mysql/
```

Notes

- You might want to set the *--use-memory* to speed up the process if you are on a dedicated server that has enough free memory. More details *here*.
- A more detailed explanation is *here*.

Restoring the Backup

Because **xtrabackup** doesn't copy *MyISAM* files, `.frm` files, and the rest of the database, you might need to back those up separately. To restore the InnoDB data, simply do something like the following after preparing:

```
cd /data/backups/mysql/
rsync -rvt --exclude 'xtrabackup_checkpoints' --exclude 'xtrabackup_logfile' \
   ./ /var/lib/mysql
chown -R mysql:mysql /var/lib/mysql/
```

4.1.3 How-Tos

How to setup a slave for replication in 6 simple steps with Percona XtraBackup

> Data is, by far, the most valuable part of a system. Having a backup done systematically and available for a rapid recovery in case of failure is admittedly essential to a system. However, it is not common practice because of its costs, infrastructure needed or even the boredom associated to the task. *Percona XtraBackup* is designed to solve this problem.
>
> You can have almost real-time backups in 6 simple steps by setting up a replication environment with *Percona XtraBackup*.

Percona XtraBackup is a tool for backing up your data extremely easy and without interruption. It performs "hot backups" on unmodified versions of *MySQL* servers (5.1, 5.5 and 5.6), as well as *MariaDB* and *Percona Servers*. It is a totally free and open source software distributed only under the *GPLv2* license.

All the things you will need

Setting up a slave for replication with *Percona XtraBackup* is really a very straightforward procedure. In order to keep it simple, here is a list of the things you need to follow the steps without hassles:

- `TheMaster` A system with a *MySQL*-based server installed, configured and running. This system will be called `TheMaster`, as it is where your data is stored and the one to be replicated. We will assume the following about this system:
  - the *MySQL* server is able to communicate with others by the standard TCP/IP port;
  - the *SSH* server is installed and configured;
  - you have a user account in the system with the appropriate permissions;
  - you have a MySQL's user account with appropriate privileges.
  - server has binlogs enabled and server-id set up to 1.
- `TheSlave` Another system, with a *MySQL*-based server installed on it. We will refer to this machine as `TheSlave` and we will assume the same things we did about `TheMaster`, except that the server-id on `TheSlave` is 2.
- `Percona XtraBackup` The backup tool we will use. It should be installed in both computers for convenience.

STEP 1: Make a backup on `TheMaster` and prepare it

At `TheMaster`, issue the following to a shell:

```
TheMaster$ innobackupex --user=yourDBuser --password=MaGiCdB1 /path/to/backupdir
```

After this is finished you should get:

```
innobackupex: completed OK!
```

This will make a copy of your *MySQL* data dir to the /path/to/backupdir/$TIMESTAMP. You have told *Percona XtraBackup* (through the **innobackupex** script) to connect to the database server using your database user and password, and do a hot backup of all your data in it (all *MyISAM*, *InnoDB* tables and indexes in them).

In order for snapshot to be consistent you need to prepare the data:

```
TheMaster$ innobackupex --user=yourDBuser --password=MaGiCdB1 /
           --apply-log /path/to/backupdir/$TIMESTAMP/
```

You need to select path where your snapshot has been taken, for example /home/backups/2012-01-16_11-14-43. If everything is ok you should get the same OK message. Now the transaction logs are applied to the data files, and new ones are created: your data files are ready to be used by the MySQL server.

Percona XtraBackup knows where your data is by reading your *my.cnf*. If you have your configuration file in a non-standard place, you should use the flag `--defaults-file=/location/of/my.cnf`.

If you want to skip writing the username/password every time you want to access *MySQL*, you can set it up in your $HOME folder. Just edit .my.cnf and add:

```
[client]
user=root
pass=MaGiCdB1
```

This is will give you root access to MySQL.

STEP 2: Copy backed up data to TheSlave

Use rsync or scp to copy the data from Master to Slave. If you're syncing the data directly to slave's data directory it's advised to stop the mysqld there.

```
TheMaster$ rsync -avprP -e ssh /path/to/backupdir/$TIMESTAMP TheSlave:/path/to/mysql/
```

After data has been copied you can back up the original or previously installed *MySQL datadir* (**NOTE**: Make sure mysqld is shut down before you move the contents of its datadir, or move the snapshot into its datadir.):

```
TheSlave$ mv /path/to/mysql/datadir /path/to/mysql/datadir_bak
```

and move the snapshot from `TheMaster` in its place:

```
TheSlave$ mv /path/to/mysql/$TIMESTAMP /path/to/mysql/datadir
```

After you copy data over, make sure *MySQL* has proper permissions to access them.

```
TheSlave$ chown mysql:mysql /path/to/mysql/datadir
```

In case the ibdata and iblog files are located in different directories outside of the datadir, you will have to put them in their proper place after the logs have been applied.

STEP 3: Configure The Master's MySQL server

Add the appropriate grant in order for slave to be able to connect to master:

```
TheMaster|mysql> GRANT REPLICATION SLAVE ON *.*  TO 'repl'@'$slaveip'
 IDENTIFIED BY '$slavepass';
```

Also make sure that firewall rules are correct and that `TheSlave` can connect to `TheMaster`. Test that you can run the mysql client on `TheSlave`, connect to `TheMaster`, and authenticate.

```
TheSlave$ mysql --host=TheMaster --user=repl --password=$slavepass
```

Verify the privileges.

```
mysql> SHOW GRANTS;
```

STEP 4: Configure The Slave's MySQL server

First copy the *my.cnf* file from `TheMaster` to `TheSlave`:

```
TheSlave$ scp user@TheMaster:/etc/mysql/my.cnf /etc/mysql/my.cnf
```

then change the following options in /etc/mysql/my.cnf:

```
server-id=2
```

and start/restart **mysqld** on `TheSlave`.

In case you're using init script on Debian based system to start mysqld, be sure that the password for `debian-sys-maint` user has been updated and it's the same as that user's password on the `TheMaster`. Password can be seen and updated in `/etc/mysql/debian.cnf`.

STEP 5: Start the replication

Look at the content of the file `xtrabackup_binlog_info`, it will be something like:

```
TheSlave$ cat /var/lib/mysql/xtrabackup_binlog_info
TheMaster-bin.000001     481
```

Execute the `CHANGE MASTER` statement on a MySQL console and use the username and password you've set up in STEP 3:

```
TheSlave|mysql> CHANGE MASTER TO
                MASTER_HOST='$masterip',
                MASTER_USER='repl',
                MASTER_PASSWORD='$slavepass',
                MASTER_LOG_FILE='TheMaster-bin.000001',
                MASTER_LOG_POS=481;
```

and start the slave:

```
TheSlave|mysql> START SLAVE;
```

STEP 6: Check

You should check that everything went OK with:

```
TheSlave|mysql> SHOW SLAVE STATUS \G
         ...
         Slave_IO_Running: Yes
         Slave_SQL_Running: Yes
         ...
         Seconds_Behind_Master: 13
         ...
```

Both `IO` and `SQL` threads need to be running. The `Seconds_Behind_Master` means the `SQL` currently being executed has a `current_timestamp` of 13 seconds ago. It is an estimation of the lag between `TheMaster` and `TheSlave`. Note that at the beginning, a high value could be shown because `TheSlave` has to "catch up" with `TheMaster`.

Adding more slaves to The Master

You can use this procedure with slight variation to add new slaves to a master. We will use *Percona XtraBackup* to clone an already configured slave. We will continue using the previous scenario for convenience but we will add `TheNewSlave` to the plot.

At `TheSlave`, do a full backup:

```
TheSlave$ innobackupex --user=yourDBuser --password=MaGiCiGaM /
          --slave-info /path/to/backupdir
```

By using the *`--slave-info`* *Percona XtraBackup* creates additional file called `xtrabackup_slave_info`.

Apply the logs:

```
TheSlave$ innobackupex --apply-log --use-memory=2G /path/to/backupdir/$TIMESTAMP/
```

Copy the directory from the `TheSlave` to `TheNewSlave` (**NOTE**: Make sure mysqld is shut down on `TheNewSlave` before you copy the contents the snapshot into its *datadir*.):

```
rsync -avprP -e ssh /path/to/backupdir/$TIMESTAMP TheNewSlave:/path/to/mysql/datadir
```

Add additional grant on the master:

```
TheMaster|mysql> GRANT REPLICATION SLAVE ON *.*  TO 'repl'@'$newslaveip'
             IDENTIFIED BY '$slavepass';
```

Copy the configuration file from `TheSlave`:

```
TheNEWSlave$ scp user@TheSlave:/etc/mysql/my.cnf /etc/mysql/my.cnf
```

Make sure you change the server-id variable in `/etc/mysql/my.cnf` to 3 and disable the replication on start:

```
skip-slave-start
server-id=3
```

After setting `server_id`, start **mysqld**.

Fetch the master_log_file and master_log_pos from the file `xtrabackup_slave_info`, execute the statement for setting up the master and the log file for `The NEW Slave`:

```
TheNEWSlave|mysql> CHANGE MASTER TO
                   MASTER_HOST='$masterip',
                   MASTER_USER='repl',
                   MASTER_PASSWORD='$slavepass',
                   MASTER_LOG_FILE='TheMaster-bin.000001',
                   MASTER_LOG_POS=481;
```

and start the slave:

```
TheSlave|mysql> START SLAVE;
```

If both IO and SQL threads are running when you check the `TheNewSlave`, server is replicating `TheMaster`.

Verifying Backups with replication and pt-checksum

One way to verify if the backup is consistent is by setting up the replication and running pt-table-checksum. This can be used to verify any type of backups, but before setting up replication, backup should be prepared and be able to run (this means that incremental backups should be merged to full backups, encrypted backups decrypted etc.).

Setting up the replication

How to setup a slave for replication in 6 simple steps with Percona XtraBackup guide provides a detailed instructions on how to take the backup and set up the replication.

For checking the backup consistency you can use either the original server where the backup was taken, or another test server created by using a different backup method (such as cold backup, mysqldump or LVM snapshots) as the master server in the replication setup.

Using pt-table-checksum

This tool is part of the *Percona Toolkit*. It performs an online replication consistency check by executing checksum queries on the master, which produces different results on replicas that are inconsistent with the master.

After you confirmed that replication has been set up successfully, you can install or download *pt-table-checksum*. This example shows downloading the latest version of *pt-table-checksum*:

```
$ wget percona.com/get/pt-table-checksum
```

Note: In order for pt-table-checksum to work correctly `libdbd-mysql-perl` will need to be installed on *Debian/Ubuntu* systems or `perl-DBD-MySQL` on *RHEL/CentOS*. If you installed the *percona-toolkit* package from the Percona repositories package manager should install those libraries automatically.

After this command has been run, *pt-table-checksum* will be downloaded to your current working directory.

Running the *pt-table-checksum* on the master will create `percona` database with the `checksums` table which will be replicated to the slaves as well. Example of the *pt-table-checksum* will look like this:

```
$ ./pt-table-checksum
    TS ERRORS  DIFFS     ROWS  CHUNKS SKIPPED    TIME TABLE
    04-30T11:31:50      0      0   633135       8       0   5.400 exampledb.aka_name
04-30T11:31:52      0      0   290859       1       0   2.692 exampledb.aka_title
Checksumming exampledb.user_info:  16% 02:27 remain
Checksumming exampledb.user_info:  34% 01:58 remain
Checksumming exampledb.user_info:  50% 01:29 remain
Checksumming exampledb.user_info:  68% 00:56 remain
Checksumming exampledb.user_info:  86% 00:24 remain
04-30T11:34:38      0      0 22187768     126       0 165.216 exampledb.user_info
04-30T11:38:09      0      0        0       1       0   0.033 mysql.time_zone_name
04-30T11:38:09      0      0        0       1       0   0.052 mysql.time_zone_transition
04-30T11:38:09      0      0        0       1       0   0.054 mysql.time_zone_transition_type
04-30T11:38:09      0      0        8       1       0   0.064 mysql.user
```

If all the values in the `DIFFS` column are 0 that means that backup is consistent with the current setup.

In case backup wasn't consistent *pt-table-checksum* should spot the difference and point to the table that doesn't match. Following example shows adding new user on the backed up slave in order to simulate the inconsistent backup:

```
mysql> grant usage on exampledb.* to exampledb@localhost identified by 'thisisnewpassword';
```

If we run the *pt-table-checksum* now difference should be spotted

```
$ ./pt-table-checksum
TS ERRORS  DIFFS     ROWS  CHUNKS SKIPPED    TIME TABLE
04-30T11:31:50      0      0   633135       8       0   5.400 exampledb.aka_name
04-30T11:31:52      0      0   290859       1       0   2.692 exampledb.aka_title
Checksumming exampledb.user_info:  16% 02:27 remain
Checksumming exampledb.user_info:  34% 01:58 remain
```

```
Checksumming exampledb.user_info:  50% 01:29 remain
Checksumming exampledb.user_info:  68% 00:56 remain
Checksumming exampledb.user_info:  86% 00:24 remain
04-30T11:34:38      0      0 22187768     126       0 165.216 exampledb.user_info
04-30T11:38:09      0      0        0       1       0   0.033 mysql.time_zone_name
04-30T11:38:09      0      0        0       1       0   0.052 mysql.time_zone_transition
04-30T11:38:09      0      0        0       1       0   0.054 mysql.time_zone_transition_type
04-30T11:38:09      1      0        8       1       0   0.064 mysql.user
```

This output shows that slave and the replica aren't in consistent state and that the difference is in the `mysql.user` table.

More information on different options that pt-table-checksum provides can be found in the *pt-table-checksum* documentation.

How to create a new (or repair a broken) GTID based slave

MySQL 5.6 introduced the new Global Transaction ID (GTID) support in replication. *Percona XtraBackup* from 2.1.0 version, automatically stores the `GTID` value in the `xtrabackup_binlog_info` when doing the backup of *MySQL* and *Percona Server* 5.6 with the `GTID` mode enabled. This information can be used to create a new (or repair a broken) `GTID` based slave.

STEP 1: Take a backup from any server on the replication environment, master or slave

Following command will take a backup to the /data/backups/$TIMESTAMP folder:

```
$ innobackupex /data/backups/
```

In the destination folder there will be a file with the name `xtrabackup_binlog_info`. This file will contain both, binary log coordinates and `GTID` information.

```
$ cat xtrabackup_binlog_info
mysql-bin.000002    1232        c777888a-b6df-11e2-a604-080027635ef5:1-4
```

That information is also printed by innobackupex after backup is taken:

```
innobackupex: MySQL binlog position: filename 'mysql-bin.000002', position 1232, gtid_executed c77788
```

STEP 2: Prepare the backup

Back will be prepared with the following command:

```
TheMaster$ innobackupex --apply-log /data/backups/$TIMESTAMP/
```

You need to select path where your snapshot has been taken, for example `/data/backups/2013-05-07_08-33-33`. If everything is ok you should get the same OK message. Now the transaction logs are applied to the data files, and new ones are created: your data files are ready to be used by the MySQL server.

STEP 3: Move the backup to the destination server

Use **rsync** or **scp** to copy the data to the destination server. If you're syncing the data directly to already running slave's data directory it's advised to stop the *MySQL* server there.

```
TheMaster$ rsync -avprP -e ssh /path/to/backupdir/$TIMESTAMP NewSlave:/path/to/mysql/
```

After you copy data over, make sure *MySQL* has proper permissions to access them.

```
NewSlave$ chown mysql:mysql /path/to/mysql/datadir
```

STEP 4: Configure and start replication

Following command will tell the new slave what was the last `GTID` executed on the master when backup was taken.

```
NewSlave > SET GLOBAL gtid_purged="c777888a-b6df-11e2-a604-080027635ef5:1-4";
NewSlave > CHANGE MASTER TO
             MASTER_HOST="$masterip",
             MASTER_USER="repl",
             MASTER_PASSWORD="$slavepass",
             MASTER_AUTO_POSITION = 1;
```

STEP 5: Check the replication status

Following command will show the slave status:

```
NewSlave > show slave status\G
         [..]
         Slave_IO_Running: Yes
         Slave_SQL_Running: Yes
         [...]
         Retrieved_Gtid_Set: c777888a-b6df-11e2-a604-080027635ef5:5
         Executed_Gtid_Set: c777888a-b6df-11e2-a604-080027635ef5:1-5
```

We can see that the slave has retrieved a new transaction with number 5, so transactions from 1 to 5 are already on this slave.

That's all, we have created a new slave in our `GTID` based replication environment.

4.1.4 Auxiliary Guides

Enabling the server to communicate via TCP/IP

Most of the Linux distributions do not enable by default to accept TCP/IP connections from outside in their MySQL or Percona Server packages.

You can check it with `netstat` on a shell:

```
$ netstat -lnp | grep mysql
tcp         0        0 0.0.0.0:3306 0.0.0.0:* LISTEN 2480/mysqld
unix 2 [ ACC ] STREAM LISTENING 8101 2480/mysqld /tmp/mysql.sock
```

You should check two things:

- there is a line starting with `tcp` (the server is indeed accepting TCP connections) and
- the first address (`0.0.0.0:3306` in this example) is different than `127.0.0.1:3306` (the bind address is not localhost's).

In the first case, the first place to look is the `my.cnf` file. If you find the option `skip-networking`, comment it out or just delete it. Also check that *if* the variable `bind_address` is set, then it shouldn't be set to localhost's but to the host's IP. Then restart the MySQL server and check it again with `netstat`. If the changes you did had no effect, then you should look at your distribution's startup scripts (like `rc.mysqld`). You should comment out flags like `--skip-networking` and/or change the `bind-address`.

After you get the server listening to remote TCP connections properly, the last thing to do is checking that the port (3306 by default) is indeed open. Check your firewall configurations (`iptables -L`) and that you are allowing remote hosts on that port (in `/etc/hosts.allow`).

And we're done! We have a MySQL server running which is able to communicate with the world through TCP/IP.

Privileges and Permissions for Users

We will be referring to "permissions" to the ability of a user to access and perform changes on the relevant parts of the host's filesystem, starting/stopping services and installing software.

By "privileges" we refer to the abilities of a database user to perform different kinds of actions on the database server.

At a system level

There are many ways for checking the permission on a file or directory. For example, `ls -ls /path/to/file` or `stat /path/to/file | grep Access` will do the job:

```
$ stat /etc/mysql | grep Access
Access: (0755/drwxr-xr-x)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2011-05-12 21:19:07.129850437 -0300
$ ls -ld /etc/mysql/my.cnf
-rw-r--r-- 1 root root 4703 Apr  5 06:26 /etc/mysql/my.cnf
```

As in this example, `my.cnf` is owned by `root` and not writable for anyone else. Assuming that you do not have `root` 's password, you can check what permissions you have on this types of files with `sudo -l`:

```
$ sudo -l
Password:
You may run the following commands on this host:
(root) /usr/bin/
(root) NOPASSWD: /etc/init.d/mysqld
(root) NOPASSWD: /bin/vi /etc/mysql/my.cnf
(root) NOPASSWD: /usr/local/bin/top
(root) NOPASSWD: /usr/bin/ls
(root) /bin/tail
```

Being able to execute with `sudo` scripts in `/etc/init.d/`, `/etc/rc.d/` or `/sbin/service` is the ability to start and stop services.

Also, If you can execute the package manager of your distribution, you can install or remove software with it. If not, having `rwx` permission over a directory will let you do a local installation of software by compiling it there. This is a typical situation in many hosting companies' services.

There are other ways for managing permissions, such as using *PolicyKit*, *Extended ACLs* or *SELinux*, which may be preventing or allowing your access. You should check them in that case.

At a database server level

To query the privileges that your database user has been granted, at a console of the server execute:

```
mysql> SHOW GRANTS;
```

or for a particular user with:

```
mysql> SHOW GRANTS FOR 'db-user'@'host';
```

It will display the privileges using the same format as for the GRANT statement.

Note that privileges may vary across versions of the server. To list the exact list of privileges that your server support (and a brief description of them) execute:

```
mysql> SHOW PRIVILEGES;
```

Installing and configuring a SSH server

Many Linux distributions only install the ssh client by default. If you don't have the ssh server installed already, the easiest way of doing it is by using your distribution's packaging system:

```
   ubuntu$ sudo apt-get install openssh-server
archlinux$ sudo pacman -S openssh
```

You may need to take a look at your distribution's documentation or search for a tutorial on the internet to configure it if you haven't done it before.

Some links of them are:

- Debian - http://wiki.debian.org/SSH
- Ubuntu - https://help.ubuntu.com/10.04/serverguide/C/openssh-server.html
- Archlinux - https://wiki.archlinux.org/index.php/SSH
- Fedora - http://docs.fedoraproject.org/en-US/Fedora/12/html/Deployment_Guide/s1-openssh-server-config.html
- CentOS - http://www.centos.org/docs/5/html/Deployment_Guide-en-US/s1-openssh-server-config.html
- RHEL - http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/6/html/Deployment_Guide/ch-OpenSSH.html

4.1.5 Assumptions in this section

Most of the times, the context will make the recipe or tutorial understandable. To assure that, a list of the assumptions, names and "things" that will appear in this section is given. At the beginning of each recipe or tutorial they will be specified in order to make it quicker and more practical.

HOST A system with a *MySQL*-based server installed, configured and running. We will assume the following about this system:

- the MySQL server is able to *communicate with others by the standard TCP/IP port*;
- a SSH server is installed and configured - see *here* if it is not;
- you have an user account in the system with the appropriate *permissions* and
- you have a MySQL's user account with appropriate *privileges*.

USER An user account in the system with shell access and appropriate permissions for the task. A guide for checking them is *here*.

DB-USER An user account in the database server with appropriate privileges for the task. A guide for checking them is *here*.

- *Recipes for xtrabackup*
- *Recipes for innobackupex*
- *How-Tos*
- *Auxiliary Guides*

CHAPTER

FIVE

MISCELLANEOUS

5.1 Frequently Asked Questions

5.1.1 Do I need an InnoDB Hot Backup license to use Percona XtraBackup?

No. Although `innobackupex` is derived from the same GPL and open-source wrapper script that InnoDB Hot Backup uses, it does not execute `ibbackup`, and the `xtrabackup` binary does not execute or link to `ibbackup`. You can use *Percona XtraBackup* without any license; it is completely separate from InnoDB Hot Backup.

5.1.2 Why is innobackupex distributed as innobackupex-1.5.1?

That's the way the source for the tool was distributed from Oracle. Nowadays, it is a symbolic link to **innobackupex** for backward compatibility with *Percona XtraBackup*.

5.1.3 What's the difference between innobackupex and innobackup?

Because **innobackupex** is a patched version of *Oracle* 's **innobackup** script (now renamed to **mysqlbackup**), it is quite similar in some ways, and familiarity with innobackup might be helpful.

Aside from the options for specific features of **innobackupex**, the main differences are:

- printing to `STDERR` instead of `STDOUT` (which enables the `--stream` option),
- the configuration file - *my.cnf* - is detected automatically (or setted with `innobackupex --defaults-file`) instead of the mandotory first argument,
- and defaults to **xtrabackup** as binary to use in the `--ibbackup`.

See *The innobackupex Option Reference* for more details.

5.1.4 Why doesn't the xtrabackup binary include MyISAM tables and other files in its backup? Should I use innobackupex always?

xtrabackup is a C program written to take advantage of the InnoDB or XtraDB features in order to make it as fast, less resource-consuming and unobtrusive (locking the database server) as possible. That is only possible with those engines. In the case of MyISAM a "read lock" is needed.

innobackupex is a script written in Perl which wraps **xtrabackup** and other backup tools (such as **tar4ibd**) and provides you with the functionality of each tool taking care of the details.

However, for some specific tasks, you must use **xtrabackup** directly. See the *Percona XtraBackup User Manual* for details on each one.

5.1.5 Are you aware of any web-based backup management tools (commercial or not) built around *Percona XtraBackup*?

Zmanda Recovery Manager is a commercial tool that uses *Percona XtraBackup* for Non-Blocking Backups:

> *"ZRM provides support for non-blocking backups of MySQL using |Percona XtraBackup|. ZRM with |Percona XtraBackup| provides resource utilization management by providing throttling based on the number of IO operations per second. |Percona XtraBackup| based backups also allow for table level recovery even though the backup was done at the database level (needs the recovery database server to be |Percona Server| with XtraDB)."*

5.1.6 xtrabackup binary fails with a floating point exception

In most of the cases this is due to not having install the required libraries (and version) by **xtrabackup**. Installing the *GCC* suite with the supporting libraries and recompiling **xtrabackup** will solve the issue. See *Compiling and Installing from Source Code* for instructions on the procedure.

5.1.7 How innobackupex handles the ibdata/ib_log files on restore if they aren't in mysql datadir?

In case the ibdata and ib_log files are located in different directories outside of the datadir, you will have to put them in their proper place after the logs have been applied.

5.1.8 Backup fails with Error 24: 'Too many open files'

This usually happens when database being backed up contains large amount of files and *Percona XtraBackup* can't open all of them to create a successful backup. In order to avoid this error the operating system should be configured appropriately so that *Percona XtraBackup* can open all its files. On Linux, this can be done with the `ulimit` command for specific backup session or by editing the `/etc/security/limits.conf` to change it globally (**NOTE**: the maximum possible value that can be set up is 1048576 which is a hard-coded constant in the Linux kernel).

5.2 *Percona XtraBackup* Release Notes

5.2.1 Older releases

Version 1.3 (unreleased)

Major changes:

- Port to *Percona Server* 5.1.47-11
- Separate into two binaries - xtrabackup for *Percona Server* and xtrabackup_50 for *MySQL* 5.x.

Fixed Bugs:

- Fixed Bug #561106: incremental crash
- Fixed duplicate `close()` problem at `xtrabackup_copy_datafile()`.

Percona *Percona XtraBackup* 1.4

Released on November 22, 2010

Percona XtraBackup version 1.4 fixes problems related to incremental backups. If you do incremental backups, it's strongly recommended that you upgrade to this release.

Functionality Added or Changed

- Incremental backups have changed and now allow the restoration of full backups containing certain rollback transactions that previously caused problems. Please see Preparing the Backups and the `--apply-log-only`. (From **innobackupex**, the `--redo-only` option should be used.) (Yasufumi Kinoshita)
  - The *Percona XtraBackup* Test Suite was implemented and is now a standard part of each distribution. (Aleksandr Kuzminsky)
- Other New Features
  - The `--prepare` now reports `xtrabackup_binlog_pos_innodb` if the information exists. (Yasufumi Kinoshita)
  - When `--prepare` is used to restore a partial backup, the data dictionary is now cleaned and contains only tables that exist in the backup. (Yasufumi Kinoshita)
  - The `--table` was extended to accept several regular expression arguments, separated by commas. (Yasufumi Kinoshita)
- Other Changes
  - Ported to the *Percona Server* 5.1.47 code base. (Yasufumi Kinoshita)
  - *Percona XtraBackup* now uses the memory allocators of the host operating system, rather than the built-in *InnoDB* allocators (see Using Operating System Memory Allocators). (Yasufumi Kinoshita)

Bugs Fixed

- Bug #595770 - XtraBack binaries are now shipped containing debug symbols by default. (Aleksandr Kuzminsky)
- Bug #589639 - Fixed a problem of hanging when tablespaces were deleted during the recovery process. (Yasufumi Kinoshita)
- Bug #611960 - Fixed a segmentation fault in **xtrabackup**. (Yasufumi Kinoshita)
- Miscellaneous important fixes related to incremental backups.

Percona XtraBackup 1.5-Beta

Released December 13, 2010 (downloads)

This release adds additional functionality to *Percona XtraBackup* 1.4, the current general availability version of *Percona XtraBackup*. This is a beta release.

Functionality Added or Changes

- Support for *MySQL* 5.5 databases has been implemented. (Yasufumi Kinoshita)

- *Percona XtraBackup* can now be built from the *MySQL* 5.1.52, *MySQL* 5.5.7, or *Percona Server* 5.1.53-12 code bases (fixes bug #683507). (Alexey Kopytov)
- The program is now distributed as three separate binaries:
  - **xtrabackup** - for use with *Percona Server* with the built-in *InnoDB* plugin
  - **xtrabackup_51** - for use with MySQL 5.0 & 5.1 with built-in *InnoDB*
  - **xtrabackup_55** - for use with *MySQL* 5.5 (this binary is not provided for the FreeBSD platform)
- Backing up only specific tables can now be done by specifying them in a file, using the `--tables-file`. (Yasufumi Kinoshita & Daniel Nichter)
- Additional checks were added to monitor the rate the log file is being overwritten, to determine if *Percona XtraBackup* is keeping up. If the log file is being overwritten faster than *Percona XtraBackup* can keep up, a warning is given that the backup may be inconsistent. (Yasufumi Kinoyasu)
- The *Percona XtraBackup* binaries are now compiled with the `-O3` **gcc** option, which may improve backup speed in stream mode in some cases.
- It is now possible to copy multiple data files concurrently in parallel threads when creating a backup, using the `--parallel` option. See The xtrabackup Option Reference and Parallel Backups. (Alexey Kopytov)

Bugs Fixed

- Bug #683507 - **xtrabackup** has been updated to build from the *MySQL* 5.1.52, *MySQL* 5.5.7, or *Percona Server* 5.1.53-12 code bases. (Alexey Kopytov)

5.2.2 *Percona XtraBackup* 1.6

Percona XtraBackup 1.6

Released on April 12, 2011 (Downloads are available here and from the Percona Software Repositories.)

Options Added

- Added option `--extra-lsndir` to **innobackupex**. When specified for the backup phase, the option is passed to **xtrabackup**, and *LSN* information is stored with the file in the specified directory. This is needed so that *LSN* information is preserved during stream backup. (Vadim Tkachenko)
- Added option `--incremental-lsn` to **innobackupex**. If specified, this option is passed directly to the **xtrabackup** binary and **–incremental-basedir** is ignored. (Vadim Tkachenko)
- Added option `--incremental-dir` to **innobackupex**. This option is passed directly to the **xtrabackup** binary. (Vadim Tkachenko)
- Added option `--safe-slave-backup` to **innobackupex**. (Daniel Nichter)
- Added option `--safe-slave-backup-timeout` to **innobackupex**. (Daniel Nichter)

Other Changes

- Eliminated some compiler warnings. (Stewart Smith)

- Ported *Percona XtraBackup* to *MySQL* 5.1.55, *MySQL* 5.5.9, *Percona Server* 5.1.55-12.6, and *Percona Server* 5.5.9-20.1 code bases. The **xtrabackup_55** binary is now based on *Percona Server* 5.5, rather than *MySQL* 5.5. Support for building against *InnoDB* plugin in *MySQL* 5.1 has been removed. (Alexey Kopytov)
- Updates were made to the built-in **innobackupex** usage docs. (Baron Schwartz, Fred Linhoss)
- Added a manual page for *Percona XtraBackup*. (Aleksandr Kuzminsky)
- Disabled auto-creating `ib_logfile*` when **innobackupex** is called with `--redo-only` or with `--incremental-dir`. If necessary `ib_logfile*` can be created later with **xtrabackup –prepare** call. (Vadim Tkachenko)
- Fixed **xtrabackup** exit code to improve portability: `EXIT_SUCCESS` on success and `EXIT_FAILURE` on a failure. (Aleksandr Kuzminsky)
- For portability, the *Percona XtraBackup* build script now tries to link with `libaio` only on Linux. (Aleksandr Kuzminsky)

Bugs Fixed

- Bug #368945 - When option `--prepare` was specified, an error message was requesting that `datadir` be set, even though it's not a required option. (Vadim Tkachenko)
- Bug #420181 - The **innobackupex** script now backs up *.CSV* tables. (Valentine Gostev)
- Bug #597384 - The `innobackup` `--include` option now handles non-*InnoDB* tables. (Vadim Tkachenko)
- Bug #606981 - Streaming *InnoDB* files with **tar4ibd** could lead to filesystem hangs when *InnoDB* was configured to access data files with the `O_DIRECT` flag. The reason was that **tar4ibd** did not have support for `O_DIRECT` and simultaneous `O_DIRECT` + non-`O_DIRECT` access to a file on Linux is disallowed. Fixed **innobackupex** and **tar4ibd** to use `O_DIRECT` on input *InnoDB* files if the value of `innodb_flush_method` is `O_DIRECT` in the *InnoDB* configuration. (Alexey Kopytov)
- Bug #646647 - Removed the bogus warning about invalid data in the Perl version string in **innobackupex**. (Baron Schwartz)
- Bug #672384 - When no log files can be found in the backup directory while executing `xtrabackup --stats`, a descriptive error message is printed instead of crashing. (Alexey Kopytov)
- Bug #688211 - Using the `--password` option with **innobackupex** to specify MySQL passwords containing special shell characters (such as "&") did not work, even when the option value was properly quoted.
- Bug #688417 - It's now possible to do incremental backups for compressed *InnoDB* tables.
- Bug #701767 - The script `innobackupex-1.5.1` was renamed to **innobackupex**. Symbolic link `innobackupex-1.5.1` was created for backupward compatibility. (Vadim Tkachenko)
- Bug #703070 - `xtrabackup_55` crashed with an assertion failure on non-Linux platforms. (Alexey Kopytov)
- Bug #703077 - Building **xtrabackup** could fail on some platforms due to an incorrect argument to `CMake`. Fixed by changing the `-DWITH_ZLIB` argument to lowercase, because that's what the `CMake` scripts actually expect. (Alexey Kopytov)
- Bug #713799 - Dropping a table during a backup process could result in assertion failure in **xtrabackup**. Now it continues with a warning message about the dropped table. (Alexey Kopytov)
- Bug #717784 - Performing parallel backups with the `--parallel` option could cause **xtrabackup** to fail with the "cannot mkdir" error. (Alexey Kopytov)

Percona XtraBackup 1.6.2

Percona is glad to announce the release of Percona XtraBackup 1.6.2 on 25 July, 2011 (Downloads are available here and from the Percona Software Repositories).

This release is purely composed of bug fixes and is the current stable release of *Percona XtraBackup*.

All of *Percona*'s software is open-source and free, all the details of the release and its development process can be found in the 1.6.2 milestone at Launchpad.

New Options

--version

The `--version` option has been added to the **xtrabackup** binary for printing its version. Previously, the version was displayed only while executing the binary without arguments or performing a backup. Bug Fixed: #610614 (Alexey Kopytov).

Changes

- As exporting tables should only be used with *innodb_file_per_table* set in the server, the variable is checked by **xtrabackup** when using the `--export` option. It will fail before applying the archived log without producing a potentially unusable backup. Bug Fixed: #758888 (Alexey Kopytov).

Bugs Fixed

- When creating an *InnoDB* with its own tablespace after taking a full backup, if the log files have been flushed, taking an incremental backup based on that full one would not contain the added table. This has been corrected by explicitly creating the tablespace before applying the delta files in such cases. Bug Fixed: #766607 (Alexey Kopytov).
- In some cases, **innobackupex** ignored the specified **xtrabackup** binary with the `--ibbackup` option. Bug Fixed: #729497 (Stewart Smith).
- Minor file descriptors leaks in error cases were fixed. Bug Fixed: #803718 (Stewart Smith).

Other Changes

- Improvements and fixes on the *Percona XtraBackup* Test Suite: #744303, #787966 < (Alexey Kopytov)
- Improvements and fixes on platform-specific distribution: #785556 (Ignacio Nin)
- Improvements and fixes on the *Percona XtraBackup* Documentation: #745185, #721339 (Rodrigo Gadea)

Percona XtraBackup 1.6.3

Percona is glad to announce the release of Percona XtraBackup 1.6.3 on 22 September, 2011 (Downloads are available here and from the *Percona Software Repositories*).

This release is purely composed of bug fixes and is the current stable release of *Percona XtraBackup*.

If the *innodb_file_per_table* server option is been used and `DDL` operations, `TRUNCATE TABLE`, `DROP/CREATE the_same_table` or `ALTER` statements on *InnoDB* tables are executed while taking a backup, an upgrade to

Percona XtraBackup 1.6.3 is **strongly recommended**. Under this scenario, if the server version is prior to 5.5.11 in 5.5 series or prior to 5.1.49 in 5.1 series, a server upgrade is also recommended.

All of *Percona* 's software is open-source and free, all the details of the release and its development process can be found in the 1.6.3 milestone at Launchpad.

Bugs Fixed

- Streaming backups did not work for compressed *InnoDB* tables due to missing support for compressed pages in **tar4ibd**. Bug Fixed: #665210 (*Alexey Kopytov*).
- *Percona XtraBackup* failed when `innodb_flush_method` in the server configuration file was set to `ALL_O_DIRECT`. Bug Fixed: #759225 (*Alexey Kopytov*).
- Due to a regression introduced in *Percona XtraBackup* 1.6.2, **innobackupex –copy-back** did not work if the **xtrabackup** binary was not specified explicitly with the `--ibbackup` option. Bug Fixed: #817132 (*Alexey Kopytov*).
- The `--slave-info` option now works correctly with `--safe-slave-backup` when either `--no-lock` or `--incremental` is also specified. Bug Fixed: #834657 (*Alexey Kopytov*).
- **tar4ibd** could fail with an error when processing doublewrite pages. Bug Fixed: #810269 (*Alexey Kopytov*).
- Unsupported command line options could cause a **tar4ibd** crash. Such options have been removed. Bug Fixed: #677279 (*Alexey Kopytov*).
- Executing `DDL` operations, `TRUNCATE TABLE`, `DROP/CREATE the_same_table` or `ALTER` statements on *InnoDB* tables while taking a backup could lead to a **xtrabackup** failure due to a tablespace `ID` mismatch when using per-table tablespaces. Note that this fix may not work correctly with *MySQL* 5.5 or *Percona Server* 5.5 prior to version 5.5.11. 5.1 releases from 5.1.49 or higher have been confirmed not to be affected. If the *innodb_file_per_table* option is been used, an upgrade to *Percona XtraBackup* 1.6.3 is **strongly recommended**. Under this scenario, if the server version is prior to 5.5.11 in 5.5 series or prior to 5.1.49 in 5.1 series, a server upgrade is also recommended. Bug Fixed: #722638 (*Alexey Kopytov*).

Other Changes

- Improvements and fixes on the *Percona XtraBackup* Test Suite: #855035, #787966 (*Alexey Kopytov*)
- Improvements and fixes on distribution: #775463, #745168, #849872, #785556 (*Ignacio Nin*)
- Improvements and fixes on the *Percona XtraBackup* Documentation: #837754, #745185, #836907 (*Rodrigo Gadea*)

Percona XtraBackup 1.6.4

Percona is glad to announce the release of Percona XtraBackup 1.6.4 on 19 December, 2011 (Downloads are available here and from the *Percona Software Repositories*).

This release is purely composed of bug fixes and is the current stable release of *Percona Percona XtraBackup*.

In this release we now compile the **xtrabackup** binary against more recent MySQL and Percona Server versions. We now build against: MySQL 5.1.59, MySQL 5.5.17, Percona Server 5.1.59-13.0 and Percona Server 5.5.16-22.0 and get all the InnoDB bug fixes each of these releases contain. Using *xtrabackup* to back up older MySQL or Percona Server releases is still supported.

This release introduces the *–rsync* option to **innobackupex**. This option is designed as an option for people experiencing problems related to **innobackupex** holding a write lock for a long time with the normal method of copying

the FRM files and non-InnoDB tables. By doing a two-phase pass over the MySQL datadir with rsync (first without a write lock and then with the write lock), we dramatically reduce the amount of time that a write lock is held. See the rsync for non-innodb files blueprint for technical implementation details.

Bugs Fixed

- **innobackupex** assumed that */usr/bin/perl* was where the Perl binary was located. With this bug fix, it instead uses */usr/bin/env perl* which fixes running of **innobackupex** on systems where Perl is not */usr/bin/perl*. Bug Fixed: #892393 (*Stewart Smith*)
- **innobackupex** reaches the server wait_timeout. This bug meant that for backups that would take a long time, **innobackupex** would hit the server wait_timeout and be disconnected, leading to a failed backup. With this bug fixed, instead of setting a large wait_timeout for the MySQL connection, **innobackupex** will regularly poll the server, keeping the connection alive while the backup is taking place. This is an important fix for backups that take a long time. Bug Fixed: #408803 (*Alexey Kopytov*)
- **innobackupex** and **xtrabackup** did not use STDOUT and STDERR conventionally. Sometimes errors would go to STDOUT and sometimes normal operating messages would go to STDERR. With this bug fixed, we have gone through both programs and ensured that only error messages go to STDERR. Bug Fixed: #514068 (*Daniel Nichter* and *Alexey Kopytov*)
- **innobackupex** would write to files named *stdout* and *stderr* to the current working directory and leave them behind. With this bug fixed, **innobackupex** will use temporary files instead of files in the current working directory. Bug Fixed: #687544 (*Valentine Gostev*)
- When a password for the MySQL connection was given to **innobackupex** with the *–password* option, **innobackupex** would log that password in plain text in the log. With this bug fixed, **innobackupex** will now just log *–password=xxxxxxxx* instead of the real password. Bug fixed #729843 (*Alexey Kopytov* and *Valentine Gostev*)
- **innobackupex** did not check that MySQL datadir was empty before *–copy-back* was run. With this bug fix, **innobackupex** will now error out of the *–copy-back* operation if the destination is not empty, avoiding potential data loss or a strang combination of a restored backup and previous data. Bug Fixed: #737569 (*Valentine Gostev*)
- **xtrabackup** would crash if the *–parallel* option was specified with a value of -1. Bug Fixed #884737 (*Alexey Kopytov*)
- The documentation for **innobackupex** (including *–help*) erroneously mentioned an *–ibbackup-binary* command line option when the option was really named *–ibbackup*. This bug fix updates the *–help* documentation for **innobackupex** to be correct. Bug Fixed: #809073 (*Alexey Kopytov*)
- There were certain situations where **innobackupex** would try to send commands to MySQL on a connection that was already closed. The primary example was when running **innobackupex** with *–incremental* and *–slave-save-info*. This bug fix simplifies the connection code so that such problems are harder to create in the future along with fixing this bug. Bug Fixed: #857788 (*Lachlan Mulcahy*)
- When copying files in stream mode, **innobackupex** does a special check that a file exists when **tar4ibd** has failed. If the file doesn't exist, it means the table was dropped while **innobackupex** was copying other files, so the error is ignored. There is a similar check when non-InnoDB files are being copied and if a table was dropped during this phase, **innobackupex** would erroneously fail with an error rather than safely ignoring the dropped table. With this bug fix, **innobackupex** now safely ignores file not found errors for non-InnoDB tables. Bug Fixed: #859546 (*Lachlan Mulcahy*)
- When the *–incremental* and *–incremental-lsn* options were specified together, **innobackupex** would give an erroneous error message when it tried to look at the contents of a directory it was yet to create. With this bug fixed, **innobackupex** will now not give that error. Bug fixed: #860133 (*Lachlan Mulcahy*)

- With the *–safe-slave-backup* option, **innobackupex** always correctly detected whether or not the host was a slave when initially deciding if it should STOP/START slave to perform a safe backup. However, in a later part of the backup, it would erroneously try to restart the slave if the host was not a slave, causing **innobackupex** to exit with a non-zero exit code even though the issue was benign. With this bug fixed, **innobackupex** will not attempt to restart the slave if the host is not a slave. Bug fixed: #860879 (*Lachlan Mulcahy*).

Percona XtraBackup 1.6.5

Percona is glad to announce the release of Percona XtraBackup 1.6.5 on 10 February, 2012 (Downloads are available here and from the *Percona Software Repositories*).

This release is purely composed of bug fixes and is the current stable release of *Percona XtraBackup*.

Bugs Fixed

- While running an incremental backup through innobackupex, you could get an error when the script was attempting to copy all the MYI/MYD/...etc files if a table was removed during the process of copying each file. A helper subroutine - copy_if_exists has been added and it is used instead. Bug fixed: #924026 (*Lachlan Mulcahy*).
- **tar4ibd** may crash on data files in a multi-file system tablespace configuration. Problem was that tar4ibd expected to read page size from the FSP header of each data file, which, in case of a multi-file system tablespace, is only available in the first file, but not in subsequent ones. That resulted in tar4ibd using a bogus page size, hence the crash. Fixed by enforcing UNIV_PAGE_SIZE as the page size for system tablespace files. Bug fixed: #891496 (*Alexey Kopytov*).
- Fix a crash when using parallel and incremental options together. **xtrabackup** function used a global buffer to store incremental page deltas. That didn't work with parallel backups. Fixed by allocating a local buffer in functions that use it. Bug fixed: #826632 (*Alexey Kopytov*).
- When preparing an incremental backups, **innobackupex** should copy all non-InnoDB files (including .frm files and non-InnoDB tables) to the full backup directory. Otherwise, any changes to .frm and/or non-InnoDB tables made between full and incremental backups lead to unusable backups. Bug fixed: #759701 (*Alexey Kopytov*).
- When using –remote-host to a non-standard SSH port, the **xtrabackup** wasn't passing the correct port to both ssh and scp, which use different options for port number (-p vs -P). Bug fixed: #733658 (*Sergei Glushchenko*).
- Unintentional change of innodb_version format in Percona Server 5.1.60. caused fatal error in **xtrabackup**. Regexps used to detect innodb_version were updated. Bug fixed: #910206 (*Alexey Kopytov*).
- When using **innobackupex** with –stream option it could place the output file in folder where non-root user does not have write access to. Bug fixed: #691090 (*Sergei Glushchenko*).
- **tar4ibd** wasn't using O_DIRECT for per-table `*.ibd` when it should. Fixed **innobackupex** to use the same tar4ibd arguments for ibdata* and `*.ibd`. Bug fixed: #925354 (*Alexey Kopytov*).
- Linux binary tarball now includes COPYING. Bug fixed: #914622 (*Ignacio Nin*).
- Fixed bug bug:*711207* **xtrabackup**: Error: write to stdout. (*Sergei Glushchenko*).

Percona XtraBackup 1.6.6

Percona is glad to announce the release of Percona XtraBackup 1.6.6 on April 4th, 2012 (Downloads are available here and from the *Percona Software Repositories*).

Option –remote-host for **innobackupex** has been deprecated in favor of the –stream option and it will be removed in future versions.

This release is purely composed of bug fixes and is the current stable release of *Percona XtraBackup*.

Bugs Fixed

- **innobackupex** now includes fast-checksums into generated my.cnf. Bug fixed #733651 (*Sergei Glushchenko*).
- In MySQL 5.1.57 a new assertion was added as a part of the fix for bug #59641. That assertion wasn't applicable when doing recovery with –apply-log-only option, and it was failing after successfully applying the log. Fix was implemented by bypassing that code. Fixed bug #938594 (*Alexey Kopytov*).
- When using parallel backup option in **xtrabackup**, backups could fail with "Operating system error number 17". Bug fixed: #900175 (*Alexey Kopytov*).

Percona XtraBackup 1.6.7

Percona is glad to announce the release of Percona XtraBackup 1.6.7 on December 20th, 2012 (Downloads are available here and from the *Percona Software Repositories*).

This release is purely composed of bug fixes and is the current stable release of *Percona XtraBackup*.

Bugs Fixed

`xtrabackup_binary` was not included in tar archive when streaming, instead it was written to the current directory. This could lead to a wrong xtrabackup binary being used when preparing backups created with the `--stream` or `--remote-host` options. Bugs fixed #723318 and #787988 (*Stewart Smith*).

`FLUSH TABLES WITH READ LOCK` was not used when creating incremental backups, which could lead to inconsistent backups when updates to non-InnoDB tables or DDL statements on any tables occurred during the backup process. Bug fixed #771981 (*Alexey Kopytov*).

Option `--safe-slave-backup` was resulting in incorrect binlog info, because in some cases innobackupex confused the response from SHOW SLAVE STATUS with the one from SHOW MASTER STATUS. Bug fixed #977101 (*Alexey Kopytov*).

innodb_data_file_path was not written to `backup-my.cnf`, this was a regression introduced in *Percona XtraBackup* 1.6.5. Bug fixed #983685 (*Sergei Glushchenko*).

Fixed spurious test suite failures with grep 2.10. Bug fixed #996483 (*Alexey Kopytov*).

When **innobackupex** was running with `--apply-log`, it was reading configuration from the server configuration file instead of `backup-my.cnf` in backup directory. Bug fixed #996493 (*Sergei Glushchenko*).

innobackupex could copy files to a wrong directory when merging an incremental backup to a full one. Bug fixed #1002688 (*Alexey Kopytov*).

Percona XtraBackup binary was leaking file descriptors on `--backup`. This was fixed by reusing the existing file descriptor so no leak occurs. Bug fixed #713267 (*Alexey Kopytov*).

Other bugs fixed: bug #1021954 (*Hrvoje Matijakovic*).

5.2.3 *Percona XtraBackup* 2.0

Percona XtraBackup 1.9.0 (2.0 BETA)

Percona is glad to announce the release of Percona XtraBackup 1.9.0 on 9th February 2012. Downloads are available from our download site here. For this BETA release, we will not be making APT and YUM repositories available, just base deb and RPM packages.

This is a *BETA* quality release and is not inteded for production. If you want a high quality, Generally Available release, you should use the current Stable version - currently 1.6.4 in the 1.6 series at the time of writing.

The 1.9.x version numbers will be used to distinguish between pre-release versions of *Percona XtraBackup* 2.0 and the Generally Available final release.

This release contains all of the features and bug fixes in *Percona XtraBackup 1.6.4*, plus the following:

New features

- *Percona XtraBackup* can now save Galera replication information while performing a backup when given the `--galera-info` option to innobackupex.
- The documentation is now bundled with *Percona XtraBackup*. It may not be included in binary packages for this beta release.
- Support for compiling and running *Percona XtraBackup* against debug versions of InnoDB. This is only for *very* advanced users.

Bugs Fixed

- `xtrabackup` will now raise an error if the transaction log wraps around before all log records are read. Previously it would print a warning and not error out, even though it would have generated an invalid backup. With this bug fix, if the log files wrap around before `xtrabackup` has read all the log records, `xtrabackup` will error out. Bug fixed: #805593 (*Alexey Kopytov*)
- MyISAM tables were backed up but not locked up during an incremental backup. Bug fixed: #771981 (*Valentine Gostev*)
- `tar4ibd` (used for streaming backups) could fail silently on backups larger than 4GB on 32bit systems. Bug fixed: #690822 (*Stewart Smith* and *Lee F*)
- `xtrabackup` ignored the `--defaults-file` option. `xtrabackup` will now fail if `--defaults-file` is not the first option on the command line. Bug fixed: #798488 (*Alexey Kopytov*)
- `xtrabackup_binary` was not included in tar archive when streaming, instead it was written to the current directory. This could cause backups with `--remote-host` to fail. Bugs Fixed: #723318 (*Alexey Kopytov*) and #787988 (*Alexey Kopytov*)
- Compiling *Percona XtraBackup* with GCC 4.6 produced compiler warnings. Bug fixed: #748064 (*Stewart Smith*)
- Improvements to incremental backups when using streaming, the addition of the `--extra-lsndir` option. Bug fixed: #680936 (*Vadim Tkachenko*)
- `innobackupex` was hardcoded to use `xtrabackup_51` for `--copy-back`. This could affect users who built from source. Bug fixed: #737462 (*Valentine Gostev*)
- If `--stats` is run without the log files properly initialised, `xtrabackup` will now print a warning instead of crashing. Bug fixed: #672384 (*Alexey Kopytov* and *Vadim Tkachenko*)

- Applying an incremental backup on a backup prepared with `--apply-log` and `--redo-only` failed to update the log files. Bug fixed: #717300 (*Valentine Gostev*, *Alexey Kopytov* and *Vadim Tkachenko*)
- Misc fixes to tests and build system: #749420, #762207, #733811, #811065

Percona XtraBackup 1.9.1 (2.0 BETA)

Percona is glad to announce the release of Percona XtraBackup 1.9.1 on 24th February 2012. Downloads are available from our download site here. For this BETA release, we will not be making APT and YUM repositories available, just base deb and RPM packages.

This is a *BETA* quality release and is not inteded for production. If you want a high quality, Generally Available release, you should use the current Stable version - currently 1.6.5 in the 1.6 series at the time of writing.

The 1.9.x version numbers will be used to distinguish between pre-release versions of *Percona XtraBackup* 2.0 and the Generally Available final release.

This release contains all of the features and bug fixes in *Percona XtraBackup 1.9.0*, plus the following:

New features

- *Percona XtraBackup* now supports compressed backups. These backups can be done in a parallel way, thus utilizing multiple CPU cores if needed. In previous versions, compression was only possible with streaming backups + external (usually single-threaded) compression utilities, which also had a number of other limitations (e.g. could not be used with parallel file copying, it was required to uncompress the entire backup to restore a single table) (*Alexey Kopytov*).
- *Percona XtraBackup* now supports streaming incremental backups. In previous versions streaming backups were performed by the innobackupex script but incremental backups were done by the xtrabackup binary which calculated deltas by scanning data files. Which meant those two feature were mutually exclusive, i.e. one couldn't do streaming incremental backups (*Alexey Kopytov*).
- As part of the backup, the LRU dump is now included as well (*Sergei Glushchenko*).

Bugs Fixed

- **tar4ibd** may crash on data files in a multi-file system tablespace configuration. Problem was that tar4ibd expected to read page size from the FSP header of each data file, which, in case of a multi-file system tablespace, is only available in the first file, but not in subsequent ones. That resulted in tar4ibd using a bogus page size, hence the crash. Bug fixed: #891496 (*Alexey Kopytov*).
- When preparing an incremental backups, **innobackupex** should copy all non-InnoDB files (including .frm files and non-InnoDB tables) to the full backup directory. Otherwise, any changes to .frm and/or non-InnoDB tables made between full and incremental backups lead to unusable backups. Bug fixed: #759701 (*Alexey Kopytov*).
- **xtrabackup** was using MySQL's datadir as it's target-dir. Target directory now defaults to the current directory, rather than MySQL's datadir. Bug fixed #489290 (*Sergei Glushchenko*).
- When using parallel backup option in **xtrabackup**, backups could fail with "Operating system error number 17". Bug fixed: #900175 (*Alexey Kopytov*).
- Regression in 2.0 branch caused "error: log block numbers mismatch". Bug fixed: #917823 (*Alexey Kopytov*).
- **xtrabackup** incremental backups didn't work with –parallel backups option. Bug fixed: #826632 (*Alexey Kopytov*).

- **innobackupex** when used for streaming backups, stored some of the files in the server's datadir, thus requiring write access to it. The fix is that it now uses tmpdir instead for streaming backups. For local ones, the backup target directory is used as before. Bug fixed: #691090 (*Sergei Glushchenko*).
- Unintentional change of innodb_version format in 5.1.60. caused fatal error in **xtrabackup**. Regexps used to detect innodb_version were updated. Bug fixed: #910206 (*Alexey Kopytov*).
- When using –remote-host to a non-standard SSH port, the **xtrabackup** wasn't passing the correct port to both ssh and scp, which use different options for port number (-p vs -P). It's now possible to pass custom SSH options to innobackupex, such as a non-standard port, with the –sshopt option. Bug fixed: #733658 (*Sergei Glushchenko*).
- While running an incremental backup through innobackupex, you could get an error when the script was attempting to copy all the MYI/MYD/...etc files if a table was removed during the process of copying each file. Bug fixed: #924026 (*Lachlan Mulcahy*).
- Fixed bug #711207 **xtrabackup**: "Error: write to stdout" (*Sergei Glushchenko*).
- Streaming incremental backups are now supported. Bug fixed: #929885 (*Alexey Kopytov*)
- A backup will now include the LRU dump for fast server startup after restore. Bug fixed: #543134 (*Sergei Glushchenko*)

Percona XtraBackup 1.9.2 (2.0 BETA)

Percona is glad to announce the release of Percona XtraBackup 1.9.2 on 28th March 2012. Downloads are available from our download site here. For this BETA release, we will not be making APT and YUM repositories available, just base deb and RPM packages.

This is a *BETA* quality release and is not inteded for production. If you want a high quality, Generally Available release, you should use the current Stable version - currently 1.6.5 in the 1.6 series at the time of writing.

The 1.9.x version numbers will be used to distinguish between pre-release versions of *Percona XtraBackup* 2.0 and the Generally Available final release.

Package name has been changed from xtrabackup to full product name, **percona-xtrabackup**.

Option –remote-host for **innobackupex** has been deprecated in favour of the –stream option and it will be removed in future versions.

This release contains all of the features and bug fixes in *Percona XtraBackup 1.9.1*, plus the following:

Bugs Fixed

- In MySQL 5.1.57 a new assertion was added as a part of the fix for bug #59641. That assertion wasn't applicable when doing recovery with –apply-log-only option, and it was failing after successfully applying the log. Fix was implemented by bypassing that code. Fixed bug #938594 (*Alexey Kopytov*).
- In some cases if *Percona XtraBackup* had discovered corruption it wouldn't say which file it is. Now it mentions the file name along with the error. Bug fixed #766033 (*Sergei Glushchenko*).
- Fixed *posix_fadvise* bug #925441 (*Alexey Kopytov*).

Percona XtraBackup 2.0.0

Percona is glad to announce the release of Percona XtraBackup 2.0.0 on 4th April 2012. Downloads are available from our download site here and *Percona Software Repositories*.

This release is the first GA (Generally Available) stable release in the 2.0 series. There have been no changes since the last pre-release (1.9.2), only the version number has changed.

This release contains all of the features and bug fixes in *Percona XtraBackup 1.9.2*.

Percona XtraBackup 2.0.1

Percona is glad to announce the release of Percona XtraBackup 2.0.1 on June 25th, 2012. Downloads are available from our download site here and *Percona Software Repositories*.

This release is the current GA (Generally Available) stable release in the 2.0 series.

Bugs Fixed

- After creating a full compressed backup, performing a compressed/uncompressed incremental backup would fail because `xtrabackup_checkpoints` was compressed. This has been fixed by omitting xtrabackup_checkpoints from compression, so that a full backup could be used for incremental backups without decompression. Bug fixed #977652 (*Alexey Kopytov*).
- `--copy-back` was copying compressed *.qp files as well. This has been fixed by skipping the compressed files while copying the data back. Bug fixed :bug:'983695' (*Alexey Kopytov*).
- Streaming backups with `--stream=tar` would fail if the file size was bigger than 8GB. Fixed by changing the libarchive format from `USTAR` to `restricted PAX` which supports bigger file sizes. Bug fixed #977998 (*Alexey Kopytov*).
- **innobackupex** was calling the tar utility unconditionally when streaming **ib_lru_dump** and **xtrabackup_galera_info**. Which led to a broken stream when the xbstream format was used. Bug fixed #983720 (*Sergei Glushchenko*).
- when `--compress` was used together with `--stream=tar`, xtrabackup was silently creating a broken backup. Now it fails with an error instead, suggesting to either use xbstream, or don't use compression at all. Bug fixed #972169 (*Alexey Kopytov*).
- `--safe-slave-backup` was resulting in incorrect binlog info, because in some cases innobackupex confused the response from SHOW SLAVE STATUS with the one from SHOW MASTER STATUS. Bug fixed #977101 (*Alexey Kopytov*).
- *xbstream* would sometimes fail while extracting the backup. Bug fixed #977995 (*Alexey Kopytov*).
- *innodb_data_file_path* was not written to backup-my.cnf, this was a regression introduced in previous version. Bug fixed #983685 (*Sergei Glushchenko*).
- *Percona XtraBackup* would fail to find the *datadir* when using `mysqld_multi`. This was fixed by adding new option `--defaults-group`, to both innobackupex and xtrabackup, now it can be specified which section of my.cnf to handle. Bug fixed #483827 (*Sergei Glushchenko* and *Daniël van Eeden*).
- InnoDB tables with names containing: *opt*, *par*, *CSV*, *MYD* were backed up twice. These tables were backed up by xtrabackup binary and by innobackupex script. Regular expression for filtering database directory contents was fixed. Bug fixed #989397 (*Sergei Glushchenko*).
- When run innobackupex with `--apply-log`, it was reading configuration from the server configuration file instead of backup-my.cnf in backup directory. Bug fixed #996493 (*Sergei Glushchenko*).
- **innobackupex** could copy files to a wrong directory when merging an incremental backup to a full one. Bug fixed #1002688 (*Alexey Kopytov*).

- Incremental backups were not working correctly with `--stream=tar`. This was fixed by making `--incremental-lsn` incompatible with `--stream=tar`. *Percona XtraBackup* will fail with an error message suggesting to use `--stream=xbstream`. Bug fixed #999750 (*Alexey Kopytov*).
- **innobackupex** failed to copy-back backup if destination dir wasn't empty. Exceptions were added for *my.cnf* and *master.info* as *Percona XtraBackup* doesn't backup those files, so it won't overwrite anything. Bug fixed #935847 (*Igor Tverdovskiy*).
- **innobackupex** `--copy-back` could skip some files when copying from a Windows filesystem mounted over NFS. Bug fixed #1003518 (*Alexey Kopytov*).
- *Percona XtraBackup* binary was leaking file descriptors on `--backup`. This was fixed by reusing the existing file descriptor so no leak occurs. Bug fixed #713267 (*Alexey Kopytov*).
- There were no source files in tar.gz archive for *Percona XtraBackup* 2.0.0. Bug fixed #1002841 (*Ignacio Nin*).
- ***Percona XtraBackup* binary could fail with the "log block checksum** mismatch" error when reading an partially written log block. Bug fixed #1015416 (*Alexey Kopytov*).

Other bugfixes: bug #970941 (*Stewart Smith*), bug #999273 (*Alexey Kopytov*) and bug #989488 (*Hrvoje Matijakovic*).

Percona XtraBackup 2.0.2

Percona is glad to announce the release of Percona XtraBackup 2.0.2 on August 13th, 2012. Downloads are available from our download site here and *Percona Software Repositories*.

This release is the current GA (Generally Available) stable release in the 2.0 series.

Bugs Fixed

- Fixed false positive test suite failures with grep 2.10. Bug fixed #996483 (*Alexey Kopytov*).
- Incremental backup would fail if a tablespace was created between full and incremental backups. Bug fixed #1022562 (*Laurynas Biveinis*).
- Assertion error in creating a compressed tablespace at delta apply time has been fixed. Bug fixed #1028949 (*Laurynas Biveinis*).
- If the table was renamed after the full backup, but before the incremental backup has been taken, incremental backups would fail when being prepared. Bug fixed #932623 (*Sergei Glushchenko*).
- When the variable *innodb_log_block_size* was set to 4096, backups would fail in the prepare stage. Bug fixed #976945 (*Sergei Glushchenko*).
- Additional incremental backup tests have been added for the incremental backup data page copy. Bug fixed #1021249 (*Laurynas Biveinis*).

Percona XtraBackup 2.0.3

Percona is glad to announce the release of Percona XtraBackup 2.0.3 on October 1st, 2012. Downloads are available from our download site here and *Percona Software Repositories*.

This release is the current GA (Generally Available) stable release in the 2.0 series.

New Features

- **innobackupex** now supports new `--move-back` option that can be used instead of `--copy-back` in case there isn't enough free disk space on the server to copy files. As this option removes backup files, it must be used with caution.

Bugs Fixed

- Symlink for `innobackupex-1.5.1` binary has been broken in the previous version of *Percona XtraBackup*. Bug fixed #1038198 (*Ignacio Nin*).
- *Percona XtraBackup* 2.0.2 was not backwards compatible which caused incremental backups created with previous versions to fail on prepare. Bug fixed #1038127 (*Sergei Glushchenko*).
- Fix for bug #1022562 introduced a regression that may potentially lead to a 5x increase in disk space occupied by incremental backups. Bug fixed #1043762 (*Laurynas Biveinis*).
- A regression was introduced in fix for bug #932623 which caused incorrect handling of compressed tablespaces with the page size of 16K, that were created between the last full or incremental and the next incremental backup. Bugs fixed #1049174 and #1044398 (*Laurynas Biveinis*).

Percona XtraBackup 2.0.4

Percona is glad to announce the release of Percona XtraBackup 2.0.4 on December 3rd, 2012. Downloads are available from our download site here and *Percona Software Repositories*.

This release is the current GA (Generally Available) stable release in the 2.0 series.

Bugs Fixed

- Bug fix for #932623 introduced the regression in *Percona XtraBackup* 2.0.2 which caused incremental backups to fail because the init parameter values were not normalized to the values used inside *InnoDB*. Bug fixed #1062684 (*Sergei Glushchenko*).
- Bug fix for #932623 introduced the regression in *Percona XtraBackup* 2.0.2 because it didn't take the separate doublewrite tablespace into an account. Bug fixed #1066843 (*Sergei Glushchenko*).
- *Percona XtraBackup* was handling the separate doublewrite buffer file incorrectly. File path of the doublewrite buffer wasn't added to the `backup-my.cnf` and after the restore old doublewrite buffer file was used instead of one made during the prepare stage. Bug fixed #1068470 (*Sergei Glushchenko*).
- *Percona XtraBackup* now accepts the `--innodb=force` option, previously it would throw an error if the option was set. Bug fixed #528752 (*Laurynas Biveinis*).
- Option `safe-slave-backup` wasn't working correctly. Bug fixed #887803 (*Alexey Kopytov*).
- In case `safe-slave-backup-timeout` was reached when using the `safe-slave-backup` option, `SQL_THREAD` was left in stopped state causing the slave thread to lag behind. This was fixed by checking the initial `SQL_THREAD` state and starting it before terminating with a timeout error and starting the `SQL_THREAD` only if it was running initially. Bug fixed #1037379 (*Alexey Kopytov*).
- *Percona XtraBackup* would fail on `--apply-log` when filesystem didn't support Linux AIO. Bug fixed #1065561 (*Alexey Kopytov*).
- `xtrabackup` binary would ignore `innodb_use_native_aio` when it's specified either in `my.cnf` or as a command line option. Bug fixed #1068459 (*Alexey Kopytov*).

- *Percona XtraBackup* would print a warning message during the prepare stage about `innodb_file_io_threads` being deprecated, even if the variable wasn't set. Bug fixed #1068485 (*Alexey Kopytov*).
- *Percona XtraBackup* Galera tests can now be run concurrently. Bug fixed #1077800 (*Stewart Smith*).

Percona XtraBackup 2.0.5

Percona is glad to announce the release of Percona XtraBackup 2.0.5 on January 18th, 2013. Downloads are available from our download site here and *Percona Software Repositories*.

This release is the current GA (Generally Available) stable release in the 2.0 series.

New Features

New option `--defaults-extra-file` has been introduced. This option specifies from what extra file to read the default *MySQL* options before the standard `defaults-file`. It can be used to load the user/password combination for the dedicated backup user from a separate configuration file, to avoid storing it in the crontab or a script somewhere in the system.

Bugs Fixed

In case of streaming backups, **innobackupex** would resume the *Percona XtraBackup* process and then wait for it to finish before running `UNLOCK TABLES`. This caused database to be unnecessarily locked with `FLUSH TABLES WITH READ LOCK`. Innobackupex now waits only till log copying is finished to unlock the databases. Bug fixed #1055989 (*Alexey Kopytov*).

innobackupex error messages referencing the data directory have been extended to show the path of the data directory mentioned in the error message. Bug fixed #1089375 (*Hartmut Holzgraefe*).

Partitioned tables were not correctly handled by the `--databases`, `--include`, `--tables-file` options of **innobackupex**, and by the `--tables` and `--tables-file` options of *Percona XtraBackup*. Fixed by removing the partition suffix (#P#...) before doing filtering. Bug fixed #711166 (*Sergei Glushchenko*).

When built-in compression was used, *Percona XtraBackup* was doing unbuffered writes to the destination file or stream in very small chunks which in return caused inefficient I/O. Fixed by using a 1M buffer for output similar to the uncompressed backups. Bug fixed #1095249 (*Alexey Kopytov*).

Unnecessary long sleep() in **innobackupex** lead to `FLUSH TABLES WITH READ LOCK` taking too long. Fixed by replacing 2 seconds sleep interval with 100 milliseconds one. Bug fixed #1095551 (*Sergei Glushchenko*).

If **innobackupex** would crash it would leave the `xtrabackup_suspended` file on the filesystem. This could then cause **innobackupex** to think *Percona XtraBackup* has suspended itself the moment it started, and then when xtrabackup actually does suspend itself innobackupex would wait for it to end and wouldn't re-remove the suspend_file, leading to a wait deadlock. Fixed by removing the stale `xtrabackup_suspended` file when **innobackupex** is started. Bug fixed #1007446 (*George Ormond Lorch III*).

innobackupex would fail to recognize MariaDB 5.2 and MariaDB 5.3. Fixed by augmenting version checks in **innobackupex**. Bug fixed #733665 (*Daniël van Eeden*, *Alexey Kopytov*).

Other bug fixes: bug fixed #924492 (*Alexey Kopytov*), bug fixed #1097158 (*Alexey Kopytov*), bug fixed #1081882 (*Alexey Kopytov*), bug fixed #1096584 (*Alexey Kopytov*),

Percona XtraBackup 2.0.6

Percona is glad to announce the release of Percona XtraBackup 2.0.6 on March 20, 2013. Downloads are available from our download site here and *Percona Software Repositories*.

This release is the current GA (Generally Available) stable release in the 2.0 series.

New Features

Percona XtraBackup has implemented basic support for *MySQL* 5.6, *Percona Server* 5.6 and *MariaDB* 10.0. Basic support means that these versions are are recognized by *Percona XtraBackup*, and that backup/restore works as long as no 5.6-specific features are used (such as `GTID`, remote/transportable tablespaces, separate undo tablespace, 5.6-style buffer pool dump files).

Bugs Fixed

Individual *InnoDB* tablespaces with size less than 1MB were extended to 1MB on the backup prepare operation. This led to a large increase in disk usage in cases when there are many small *InnoDB* tablespaces. Bug fixed #950334 (*Daniel Frett*, *Alexey Kopytov*).

Fixed the issue that caused databases corresponding to inaccessible *datadir* subdirectories to be ignored by *Percona XtraBackup* without warning or error messages. This was happening because *InnoDB* code silently ignored *datadir* subdirectories it could not open. Bug fixed #664986 (*Alexey Kopytov*).

Under some circumstances *Percona XtraBackup* could fail to copy a tablespace with a high `--parallel` option value and a low `innodb_open_files` value. Bug fixed #870119 (*Alexey Kopytov*).

Fix for the bug #711166 introduced a regression that caused individual partition backups to fail when used with `--include` option in **innobackupex** or the `--tables` option in xtrabackup. Bug fixed #1130627 (*Alexey Kopytov*).

innobackupex didn't add the `file-per-table` setting for table-independent backups. Fixed by making *Percona XtraBackup* auto-enable `innodb_file_per_table` when the `--export` option is used. Bug fixed #930062 (*Alexey Kopytov*).

Under some circumstances *Percona XtraBackup* could fail on a backup prepare with `innodb_flush_method=O_DIRECT`. Bug fixed #1055547 (*Alexey Kopytov*).

innobackupex did not pass the `--tmpdir` option to the xtrabackup binary resulting in the server's `tmpdir` always being used for temporary files. Bug fixed #1085099 (*Alexey Kopytov*).

Percona XtraBackup has improved the error reporting for unrecognized server versions. Bug fixed #1087219 (*Alexey Kopytov*).

Fixed the missing `rpm` dependency for Perl `Time::HiRes` package that caused **innobackupex** to fail on minimal CentOS installations. Bug fixed #1121573 (*Alexey Bychko*).

innobackupex would fail when `--no-lock` and `--rsync` were used in conjunction. Bug fixed #1123335 (*Sergei Glushchenko*).

Fix for the bug #1055989 introduced a regression that caused `xtrabackup_pid` file to remain in the temporary dir after execution. Bug fixed #1114955 (*Alexey Kopytov*).

Unnecessary debug messages have been removed from the *Percona XtraBackup* output. Bug fixed #1131084 (*Alexey Kopytov*).

Other bug fixes: bug fixed #1153334 (*Alexey Kopytov*), bug fixed #1098498 (*Laurynas Biveinis*), bug fixed #1132763 (*Laurynas Biveinis*), bug fixed #1142229 (*Laurynas Biveinis*), bug fixed #1130581 (*Laurynas Biveinis*).

Percona XtraBackup 2.0.7

Percona is glad to announce the release of Percona XtraBackup 2.0.7 on May 6, 2013. Downloads are available from our download site here and *Percona Software Repositories*.

This release is the current GA (Generally Available) stable release in the 2.0 series.

New Features

This version of *Percona XtraBackup* has implemented full support for new *MySQL* 5.6 features (`GTID`, remote/transportable tablespaces, separate undo tablespace, 5.6-style buffer pool dump files).

Percona XtraBackup has implemented support for the InnoDB Buffer Pool Preloading introduced in *MySQL* 5.6. Starting with *MySQL* 5.6 buffer pool dumps can be produced and loaded for faster server warmup after the start. This feature is similar to the Dump/Restore of the Buffer Pool in *Percona Server*. *MySQL* 5.6 buffer pool dump is copied into backup directory during the backup stage. During the copy back stage (restore) it is copied back to data directory. After the backup is restored buffer pool dump can be loaded by the server either automatically on startup or on demand.

Time interval between checks done by log copying thread is now configurable by `innobackupex --log-copy-interval`. Making the interval configurable allows to reduce the time between checks which can prevent *Percona XtraBackup* failures that are caused by the log records in the transactional log being overwritten before they are copied by the log copying thread.

Percona XtraBackup now stores the `GTID` value in the `xtrabackup_binlog_info` when doing the backup of *MySQL* and *Percona Server* 5.6 with the `GTID` mode enabled. Example of how this information can be used to create/restore a slave can be found in this blogpost.

Percona XtraBackup option `xtrabackup --export` now supports transportable tablespaces introduced in *MySQL* 5.6. This option can be used to produce 5.6-style metadata files, that can be imported by `ALTER TABLE IMPORT TABLESPACE` on *MySQL* and *Percona Server* 5.6 as described in *Restoring Individual Tables* guide.

Bugs Fixed

`xtrabackup_56` binary was present in `rpm` and `deb` packages, but it was missing from the source `.tar.gz` package. Fixed by adding the missing binary to `.tar.gz` as well. Bug fixed #1158948.

innobackupex could crash when taking the 5.6 backup due to linking the wrong `SSL` library. Bug fixed #1168540.

Percona XtraBackup would crash when preparing the 5.6 backup with partitioned tables. Bug fixed #1169169.

Tables that were dropped between taking a full backup and an incremental one were present in the full backup directory, and were not removed when incremental backups has been merged. Fixed by removing files corresponding to tables that are missing in the incremental backup directory. Bug fixed #856400.

Percona XtraBackup would leave stale `xtrabackup_tmp*` files in the *datadir* after applying incremental backups. Bug fixed #1079135.

Fixed couple of warnings found in **innobackupex** when all warnings have been made `FATAL`. Bug fixed #1116177.

If there are thousands of tables and slow IO then *Percona XtraBackup* can spend a lot of time opening all the tablespaces. Optimization has been implemented and *Percona XtraBackup* now avoids loading non-relevant tablespaces when partial backup is being taken which speeds up the backup process. Bug fixed #1130145.

Percona XtraBackup didn't initialize per-thread data in the log copying thread which could cause *Percona XtraBackup* to crash. Bug fixed #1166888.

Package dependency has been changed from abstract `mysql` to real `/usr/bin/mysql` file, because `rpm` packages from *Oracle* no longer satisfied `mysql` dependency which is required by the *Percona XtraBackup* `rpms`. Bug fixed #1095972.

Percona XtraBackup would fail when preparing the *MySQL* 5.6 backup if the log files were bigger than 4G on the source server. Bug fixed #1164979.

Due to different implementation in *MySQL* 5.6 error messages were not printed to `stderr` directly. Because of that all InnoDB error or diagnostic messages are never printed by xtrabackup_56. Bug fixed #1169971.

innobackupex would still run with `FLUSH TABLES WITH READ LOCK` even if **xtrabackup** would fail when copying logs. Fixed by terminating **xtrabackup** process immediately on log copying failure. Bug fixed #1170806.

innobackupex would fail if the `SQL_MODE` was set to `ANSI_QUOTES`. Bug fixed #945161.

Missing space_id from `*.ibd.meta` would lead to assertion. Fixed by replacing the assertion with the error message. Bug fixed #1112224.

Fixed the typo in the **innobackupex** error output. Bug fixed #1157225.

When building from source `innodb56` target didn't have an option to disable DTrace like `innodb55` has. Fixed by adding `-DENABLE_DTRACE=OFF` build option for `innodb56` as well. Bug fixed #1169509.

innobackupex wasn't handling the *`innodb_data_file_path`* option which could cause backup to fail. Bug fixed #1169726.

For the *Debian* and the *Linux* binaries, the *`--version`* message which should include the revision was showing "undefined". Bug fixed #1171721.

Redundant code has been removed from `xtrabackup.cc`. Bug fixed #1162765.

Other bug fixes: bug fixed #1158154, bug fixed #1170340, bug fixed #1088309, bug fixed #1088307.

Percona XtraBackup 2.0.8

Percona is glad to announce the release of Percona XtraBackup 2.0.8 on September 4th, 2013. Downloads are available from our download site here and *Percona Software Repositories*.

This release is the current GA (Generally Available) stable release in the 2.0 series.

Bugs Fixed

Percona XtraBackup 2.0 will now be offered in *Percona Software Repositories*. Bug fixed #1190055.

Percona XtraBackup would assume the table has been dropped if the tablespace was renamed after it was scanned by *Percona XtraBackup* on startup and before *Percona XtraBackup* attempted to copy it. Bug fixed #1079700.

Fixed the `libssl.so.6` dependency issues in binary tarballs releases. Bug fixed #1172916.

Orphaned `xtrabackup_pid` file left inside tmpdir could cause SST to fail. Fixed by fix checking if `xtrabackup_pid` file exists once **innobackupex** starts, and try to remove it or fail if it cannot be removed. Bug fixed #1175860.

During the backup process loading tablespaces was started before the log copying, this could lead to a race between the datafiles state in the resulting backup and xtrabackup_logfile. Tablespace created at a sensitive time would be missing in both the backup itself and as the corresponding log record in `xtrabackup_logfile`, so it would not be created on *`innobackupex --apply-log`* either. Bug fixed #1177206.

innobackupex automatic version detection did not work correctly for latest *Percona Server* and *MySQL* 5.1 releases which could cause innobackupex to fail. Bugs fixed #1181092 and #1181099.

Difference in behavior between *InnoDB* 5.5 and 5.6 codebases in cases when a newly created tablespace has uninitialized first page at the time when *Percona XtraBackup* opens it while creating a list of tablespaces to backup would cause assertion error. Bug fixed #1187071.

Debug builds would fail due to compiler errors on *Ubuntu* Quantal/Raring builds. Fixed compiler warnings by backporting the corresponding changes from upstream. Bug fixed #1192454.

Percona XtraBackup would stop in case log block numbers had to wrap around, which only happens once per 1 GB of log writes, and the wrap-around point was between the last checkpoint and the current log tail at the time the backup starts. Bug fixed #1206309.

`xtrabackup_56` binary would fail to create a suspend file, which would result in an error. Bug fixed #1210266.

Under some circumstances *Percona XtraBackup* could fail on a backup prepare with `innodb_flush_method=O_DIRECT` when `XFS` filesystem was being used. Bug fixed #1190779.

Percona XtraBackup didn't recognize checkpoint `#0` as a valid checkpoint on *`xtrabackup --prepare`* which would cause an error. Bug fixed #1196475.

`xtrabackup --stats` option would not work with server *datadir* if the server isn't running and logs were in a separate directory. Bug fixed #1174314.

Other bug fixes: bug fixed #1097434, bug fixed #1214272, bug fixed #1211173, bug fixed #1201599, bug fixed #1097444, bug fixed #1042796, bug fixed #1214730, bug fixed #1204463, bug fixed #1197249, bug fixed #1196894, bug fixed #1194813, bug fixed #1183500, bug fixed #1177182, bug fixed #1175309, bug fixed #1201686, bug fixed #1182995, bug fixed #1175566.

5.2.4 Percona *Percona XtraBackup* 2.1

Percona XtraBackup 2.1.0-alpha1

Percona is glad to announce the release of *Percona XtraBackup* 2.1.0-alpha1 on April 2nd 2013. Downloads are available from our download site here. For this ALPHA release, we will not be making APT and YUM repositories available, just base deb and RPM packages

This is an *ALPHA* quality release and is not intended for production. If you want a high quality, Generally Available release, the current Stable version should be used (currently 2.0.6 in the 2.0 series at the time of writing).

This release contains all of the features and bug fixes in *Percona XtraBackup 2.0.6*, plus the following:

New features

Percona XtraBackup now has support for *Compact Backups*. This feature can be used for taking the backups that will take less amount of disk space.

Percona XtraBackup has implemented *Encrypted Backups*. This feature can be used to encrypt/decrypt both local and streamed backups in order to add another layer of protection to the backups.

innobackupex now uses Perl's `DBD::MySQL` package for server communication instead of spawning the mysql command line client.

Support for *InnoDB* 5.0 and *InnoDB* 5.1 builtin has been removed from *Percona XtraBackup*.

After being deprecated in previous version, option `--remote-host` has been completely removed in *Percona XtraBackup* 2.1.

Bugs Fixed

innobackupex now supports empty arguments in the `--password` option. Bug fixed #1032667 (*Andrew Gaul*).

Percona XtraBackup 2.1.0-beta1

Percona is glad to announce the release of *Percona XtraBackup* 2.1.0-beta1 on April 22nd 2013. Downloads are available from our download site here. For this BETA release, we will not be making APT and YUM repositories available, just base deb and RPM packages

This is an *BETA* quality release and is not intended for production. If you want a high quality, Generally Available release, the current Stable version should be used (currently 2.0.6 in the 2.0 series at the time of writing).

This release contains all of the features and bug fixes in *Percona XtraBackup 2.0.6*, plus the following:

New features

Percona XtraBackup has implemented basic support for *MySQL* 5.6, *Percona Server* 5.6 and *MariaDB* 10.0. Basic support means that these versions are are recognized by *Percona XtraBackup*, and that backup/restore works as long as no 5.6-specific features are used (such as `GTID`, remote/transportable tablespaces, separate undo tablespace, 5.6-style buffer pool dump files).

Percona XtraBackup can use XtraDB changed page tracking feature to perform the *Incremental Backups* now.

Bugs Fixed

Fixed couple of warnings found in **innobackupex** when all warnings have been made `FATAL`. Bug fixed #1116177.

innobackupex is using `SHOW MASTER STATUS` to obtain binlog file and position. This could trigger a bug if the server being backed up was standalone server (neither master nor slave in replication) and binlog information wasn't available. Fixed by not creating `xtrabackup_binlog_info` file when binlog isn't available. Bug fixed #1168513.

Fixed the typo in the **innobackupex** error output. Bug fixed #1157225.

Redundant code has been removed from `xtrabackup.cc`. Bug fixed #1162765.

Other bugs fixed: bug fixed #1158154, bug fixed #1166713.

Percona XtraBackup 2.1.0-rc1

Percona is glad to announce the release of *Percona XtraBackup* 2.1.0-rc1 on May 7th 2013. Downloads are available from our download site here. For this *RC* release, we will not be making APT and YUM repositories available, just base deb and RPM packages.

This is an *Release Candidate* quality release and is not intended for production. If you want a high quality, Generally Available release, the current Stable version should be used (currently 2.0.7 in the 2.0 series at the time of writing).

New features

This version of *Percona XtraBackup* has implemented full support for new *MySQL* 5.6 features (`GTID`, remote/transportable tablespaces, separate undo tablespace, 5.6-style buffer pool dump files).

Percona XtraBackup has implemented support for the InnoDB Buffer Pool Preloading introduced in *MySQL* 5.6. Starting with *MySQL* 5.6 buffer pool dumps can be produced and loaded for faster server warmup after the start. This feature is similar to the Dump/Restore of the Buffer Pool in *Percona Server*. *MySQL* 5.6 buffer pool dump is copied into backup directory during the backup stage. During the copy back stage (restore) it is copied back to data directory. After the backup is restored buffer pool dump can be loaded by the server either automatically on startup or on demand.

Time interval between checks done by log copying thread is now configurable by *`innobackupex --log-copy-interval`*. Making the interval configurable allows to reduce the time between checks which can prevent *Percona XtraBackup* failures that are caused by the log records in the transactional log being overwritten before they are copied by the log copying thread.

Percona XtraBackup now stores the `GTID` value in the `xtrabackup_binlog_info` when doing the backup of *MySQL* and *Percona Server* 5.6 with the `GTID` mode enabled. Example of how this information can be used to create/restore a slave can be found in this blogpost.

Percona XtraBackup option *`xtrabackup --export`* now supports transportable tablespaces introduced in *MySQL* 5.6. This option can be used to produce 5.6-style metadata files, that can be imported by `ALTER TABLE IMPORT TABLESPACE` on *MySQL* and *Percona Server* 5.6 as described in *Restoring Individual Tables* guide.

Bugs Fixed

Percona XtraBackup would crash when preparing the 5.6 backup with partitioned tables. Bug fixed #1169169.

Tables that were dropped between taking a full backup and an incremental one were present in the full backup directory, and were not removed when incremental backups has been merged. Fixed by removing files corresponding to tables that are missing in the incremental backup directory. Bug fixed #856400.

Percona XtraBackup would leave stale `xtrabackup_tmp*` files in the *datadir* after applying incremental backups. Bug fixed #1079135.

If there are thousands of tables and slow IO then *Percona XtraBackup* can spend a lot of time opening all the tablespaces. Optimization has been implemented and *Percona XtraBackup* now avoids loading non-relevant tablespaces when partial backup is being taken which speeds up the backup process. Bug fixed #1130145.

Due to different implementation in *MySQL* 5.6 error messages were not printed to `stderr` directly. Because of that all InnoDB error or diagnostic messages are never printed by xtrabackup_56. Bug fixed #1169971

innobackupex would still run with `FLUSH TABLES WITH READ LOCK` even if **xtrabackup** would fail when copying logs. Fixed by terminating **xtrabackup** process immediately on log copying failure. Bug fixed #1170806.

Percona XtraBackup would leave `xbtemp` temp files behind due to a typo. Bug fixed #1172016.

innobackupex wasn't handling the *innodb_data_file_path* option which could cause backup to fail. Bug fixed #1169726.

For the *Debian* and the *Linux* binaries, the *--version* message which should include the revision was showing "undefined". Bug fixed #1171721.

Other bugs fixed: bug fixed #1088307, bug fixed #1088309, bug fixed #1170340.

Percona XtraBackup 2.1.1

Percona is glad to announce the release of *Percona XtraBackup* 2.1.1 on May 15th 2013. Downloads are available from our download site here and *Percona Software Repositories*.

This release is the first GA (Generally Available) stable release in the 2.1 series.

New features

Percona XtraBackup now has support for *Compact Backups*. This feature can be used for taking the backups that will take less amount of disk space. GA release now contains new *innobackupex --rebuild-threads* that can be used to specify the number of threads started by XtraBackup when rebuilding secondary indexes on `innobackupex --apply-log --rebuild-indexes`. This allows parallel processing of individual tables when rebuilding the index.

Percona XtraBackup has implemented *Encrypted Backups*. This feature can be used to encrypt/decrypt both local and streamed backups in order to add another layer of protection to the backups.

innobackupex now uses Perl's `DBD::MySQL` package for server communication instead of spawning the mysql command line client.

Support for *InnoDB* 5.0 and *InnoDB* 5.1 builtin has been removed from *Percona XtraBackup*.

After being deprecated in previous version, option *--remote-host* has been completely removed in *Percona XtraBackup* 2.1.

Percona XtraBackup can use XtraDB changed page tracking feature to perform the *Incremental Backups* now.

Bugs Fixed

innobackupex is using `SHOW MASTER STATUS` to obtain binlog file and position. This could trigger a bug if the server being backed up was standalone server (neither master nor slave in replication) and binlog information wasn't available. Fixed by not creating `xtrabackup_binlog_info` file when binlog isn't available. Bug fixed #1168513.

Percona XtraBackup would leave `xbtemp` temp files behind due to a typo. Bug fixed #1172016.

Percona XtraBackup would assume the table has been dropped if the tablespace was renamed after it was scanned by *Percona XtraBackup* on startup and before *Percona XtraBackup* attempted to copy it. Bug fixed #1079700.

Orphaned `xtrabackup_pid` file left inside tmpdir could cause SST to fail. Fixed by fix checking if `xtrabackup_pid` file exists once **innobackupex** starts, and try to remove it or fail if it cannot be removed. Bug fixed #1175860.

`xtrabackup --stats` option would not work with server *datadir* if the server isn't running and logs were in a separate directory. Bug fixed #1174314.

Other bugs fixed: bug fixed #1166713, bug fixed #1175581, bug fixed #1175318, bug fixed #1175309, bug fixed #1176198, bug fixed #1175566.

Percona XtraBackup 2.1.2

Percona is glad to announce the release of *Percona XtraBackup* 2.1.2 on May 17th 2013. Downloads are available from our download site here and *Percona Software Repositories*.

This release fixes number of high priority bugs since 2.1 became GA. It's advised to upgrade your latest 2.1 version to 2.1.2. This release is the latest stable release in the 2.1 series.

Bugs Fixed

Using Perl's `DBD::MySQL` package for server communication instead of spawning the mysql command line client introduced a regression which caused *`innobackupex --galera-info`* to fail. Bug fixed #1180672.

The format of `xtrabackup_galera_info` was missing the ':' separator between the values of *`wsrep_local_state_uuid`* and *`wsrep_last_committed`*. Bug fixed #1181222.

innobackupex automatic version detection did not work correctly for latest *Percona Server* and *MySQL* 5.1 releases which could cause innobackupex to fail. Bugs fixed #1181092, #1181099 and #1180905.

When backing up a server that is not a replication slave with the *`innobackupex --slave-info`* option, **innobackupex** failed with a fatal error. Replaced the fatal error with a diagnostic message about *`innobackupex --slave-info`* being ignored in such a case. Bug fixed #1180662.

Low values for `wait_timeout` on the server could cause server to close the connection while backup is being taken. Fixed by setting the bigger value for `wait_timeout` option on the server to prevent server from closing connections if the global `wait_timeout` value is set too low. Bug fixed #1180922.

Other bug fixes: bug fixed #1177182.

Percona XtraBackup 2.1.3

Percona is glad to announce the release of *Percona XtraBackup* 2.1.3 on May 22nd 2013. Downloads are available from our download site here and *Percona Software Repositories*.

This release fixes a high priority bug. It's advised to upgrade your latest 2.1 version to 2.1.3 if you're using the *Percona XtraBackup* with *Percona XtraDB Cluster*. This release is the latest stable release in the 2.1 series.

Bugs Fixed

Percona XtraBackup 2.1.2 would hang when performing State Snapshot Transfer. Bug fixed #1182698.

Percona XtraBackup 2.1.4

Percona is glad to announce the release of *Percona XtraBackup* 2.1.4 on August 8th 2013. Downloads are available from our download site here and *Percona Software Repositories*.

New Features

Percona XtraBackup has introduced *additional options* to handle the locking during the `FLUSH TABLES WITH READ LOCK`. These options can be used to minimize the amount of the time when *MySQL* operates in the read-only mode.

Percona XtraBackup has now been rebased on *MySQL* versions 5.1.70, 5.5.30, 5.6.11 and *Percona Server* versions 5.1.70-rel14.8 and 5.5.31-rel30.3 server versions.

In order to speed up the backup process, slave thread is not stopped during copying non-InnoDB data when *`innobackupex --no-lock`* option is used as using this option requires absence of `DDL` or `DML` to non-transaction tables during backup.

Source tarball (and Debian source) now include all *MySQL* source trees required for the build. This means internet connection during package build isn't required anymore.

Two new options options, *`innobackupex --decrypt`* and *`innobackupex --decompress`*, have been implemented to make *decryption* and *decompression* process more user friendly.

Bugs Fixed

There were no 2.1.x release packages available for *Ubuntu Raring*. Bug fixed #1199257.

During the backup process loading tablespaces was started before the log copying, this could lead to a race between the datafiles state in the resulting backup and xtrabackup_logfile. Tablespace created at a sensitive time would be missing in both the backup itself and as the corresponding log record in `xtrabackup_logfile`, so it would not be created on *`innobackupex --apply-log`* either. Bug fixed #1177206.

Fixed the `libssl.so.6` dependency issues in binary tarballs releases. Bug fixed #1172916.

innobackupex did not encrypt non-InnoDB files when doing local (i.e. non-streaming) backups. Bug fixed #1160778.

Difference in behavior between *InnoDB* 5.5 and 5.6 codebases in cases when a newly created tablespace has uninitialized first page at the time when *Percona XtraBackup* opens it while creating a list of tablespaces to backup would cause assertion error. Bug fixed #1187071.

xbcrypt could sometimes fail when reading encrypted stream from a pipe or network. Bug fixed #1190610.

innobackupex could not prepare the backup if there was no `xtrabackup_binary` file in the backup directory and the xtrabackup binary was not specified explicitly with *`innobackupex --ibbackup`* option. Bug fixed #1199190.

Debug builds would fail due to compiler errors on *Ubuntu* Quantal/Raring builds. Fixed compiler warnings by backporting the corresponding changes from upstream. Bug fixed #1192454.

innobackupex would terminate with an error if *`innobackupex --safe-slave-backup`* option was used for backing up the master server. Bug fixed #1190716.

Under some circumstances *Percona XtraBackup* could fail on a backup prepare with `innodb_flush_method=O_DIRECT` when `XFS` filesystem was being used. Bug fixed #1190779.

Percona XtraBackup didn't recognize checkpoint `#0` as a valid checkpoint on `xtrabackup --prepare` which would cause an error. Bug fixed #1196475.

Percona XtraBackup didn't recognize the `O_DIRECT_NO_FSYNC` value for `innodb_flush_method` which was introduced in *MySQL* 5.6.7. Fixed by adding the value to the list of supported values for `innodb_flush_method` in `xtrabackup_56`. Bug fixed #1206363.

innobackupex would terminate if `innobackupex --galera-info` option was specified when backing up non-galera server. Bug fixed #1192347.

Other bug fixes: bug fixed #1097434, bug fixed #1201599, bug fixed #1198220, bug fixed #1097444, bug fixed #1042796, bug fixed #1204463, bug fixed #1197644, bug fixed #1197249, bug fixed #1196894, bug fixed #1194813, bug fixed #1183500, bug fixed #1181432, bug fixed #1201686, bug fixed #1182995, bug fixed #1204085, bug fixed #1204083, bug fixed #1204075, bug fixed #1203672, bug fixed #1190876, bug fixed #1194879, bug fixed #1194837.

Known Issues

Backups of *MySQL* / *Percona Server* 5.6 versions prior to 5.6.11 cannot be prepared with *Percona XtraBackup* 2.1.4. Until the upstream bug #69780 is fixed and merged into *Percona XtraBackup*, *Percona XtraBackup* 2.1.3 should be used to prepare and restore such backups. This issue is reported as bug #1203669.

Percona XtraBackup 2.1.5

Percona is glad to announce the release of *Percona XtraBackup* 2.1.5 on September 19th 2013. Downloads are available from our download site here and *Percona Software Repositories*.

This release is the current GA (Generally Available) stable release in the 2.1 series.

New Features

Percona XtraBackup now supports new form of *incremental backups* for *Percona Server* 5.6 that uses Log Archiving for XtraDB feature.

Percona XtraBackup now supports new `innobackupex --version-check` option. When specified, **innobackupex** will perform a version check against the server on the backup stage after creating a server connection.

Bugs Fixed

Percona XtraBackup did not close temporary files created when preparing a compact backup, which would lead to excessive disk space usage until the prepare process finished. Bug fixed #1111380.

Depending on the subroutine **innobackupex** could sometimes leave the child processes running in case of the error. **innobackupex** now makes sure that all child processes are killed if an error occurs in the script. Bug fixed #1135441.

The 5.6-based binary (`xtrabackup_56`), which is used to backup both *MySQL* 5.6 and *Percona Server* 5.6 servers, did not support Percona Server-specific `innodb_log_block_size` option in Percona Server 5.6.11+ and would fail when trying to backup a server with a non-default `innodb_log_block_size` value. Bug fixed #1194828.

Percona XtraBackup would stop in case log block numbers had to wrap around, which only happens once per 1 GB of log writes, and the wrap-around point was between the last checkpoint and the current log tail at the time the backup starts. Bug fixed #1206309.

`xtrabackup_56` binary would fail to create a suspend file, which would result in an error. Bug fixed #1210266.

Regression was introduced in *Percona XtraBackup* 2.1.4 which lead to `cp` utility being used to copy metadata files even if the *innobackupex --rsync* option was used. Bug fixed #1211263.

Other bugs fixed: bug fixed #1214272, bug fixed #1214730, bug fixed #1213102, bug fixed #1213036, bug fixed #1204045, bug fixed #1154476, bug fixed #1195402, bug fixed #1195055.

Percona XtraBackup 2.1.6

Percona is glad to announce the release of *Percona XtraBackup* 2.1.6 on November 25th 2013. Downloads are available from our download site here and *Percona Software Repositories*.

This release is the current GA (Generally Available) stable release in the 2.1 series.

New Features

Percona XtraBackup now supports logs created with the new log block checksums option innodb_log_checksum_algorithm in *Percona Server* 5.6

New *innobackupex --force-non-empty-directories* option has been implemented. When specified, it makes *innobackupex --copy-back* option or *innobackupex --move-back* option transfer files to non-empty directories. No existing files will be overwritten. If *--copy-back* or *--move-back* has to copy a file from the backup directory which already exists in the destination directory, it will still fail with an error.

Bugs Fixed

innobackupex --copy-back would fail if *innodb_data_home_dir* is empty. Bug fixed #1049291.

A fixed initialization vector (constant string) was used while encrypting the data. This opened the encrypted stream/data to plaintext attacks among others. Bug fixed #1185343.

innobackupex --version-check is now on by default. Bug fixed #1227988.

`xtrabackup_slave_info` didn't contain any `GTID` information, which could cause `master_auto_position` not to work properly. Bug fixed #1239670.

`xtrabackup_56` was using `CRC32` as the default checksum algorithm. This could cause error if the innodb_checksum_algorithm value was changed to `strict_innodb` value after a restore. Bug fixed #1247586.

`xtrabackup_56` binary didn't store the server's innodb_checksum_algorithm value to `backup-my.cnf`. This value is needed because it affects the on-disk data format. Bug fixed #1248065.

Since Version Check is enabled by default in *Percona XtraBackup* 2.1.6, new *innobackupex --no-version-check* option has been introduced to disable it. Bug fixed #1248900.

Percona XtraBackup now supports absolute paths in *innodb_data_file_path* variable. Bug fixed #382742.

innobackupex wasn't able to perform backups to the `NFS` mount in some `NFS` configurations, because it was trying to preserve file ownership. Bug fixed #943750.

Percona XtraBackup wouldn't back up the empty directory created with `mkdir` (i.e. `test`) outside of the server which could lead to inconsistencies during the *Percona XtraDB Cluster* State Snapshot Transfer. Bug fixed #1217426.

If the innodb_log_arch_dir variable was specified in the *Percona Server* configuration file `my.cnf` *Percona XtraBackup* was unable to perform the backup. Bug fixed #1227240.

Race condition in `start_query_killer` child code could cause parent *MySQL* connection to close. Bug fixed #1239728.

Other bugs fixed: #1248488, #1247057, #1250738, #1214274.

Percona XtraBackup 2.1.7

Percona is glad to announce the release of *Percona XtraBackup* 2.1.7 on January 24th 2014. Downloads are available from our download site here and *Percona Software Repositories*.

This release is the current GA (Generally Available) stable release in the 2.1 series.

New Features

Percona XtraBackup has now been rebased on *MySQL* versions 5.1.73, 5.5.35, 5.6.15 and *Percona Server* versions 5.1.73-rel14.11 and 5.5.35-rel33.0.

Bugs Fixed

After being rebased on *MySQL* 5.6.11 *Percona XtraBackup* has been affected by the upstream bug #69780. Fixed by rebasing *Percona XtraBackup* on *MySQL* 5.6.15 which contains a fix for the upstream bug. Bug fixed #1203669.

Backup directory would need to be specified even for running the **innobackupex** with *`innobackupex --help`* and *`innobackupex --version`* options. Bug fixed #1223716.

When creating an incremental backup with the changed page tracking feature available in *Percona Server*, **innobackupex** would fail if the server had the `ANSI_QUOTES` SQL mode enabled. Bug fixed #1248331.

When *`innobackupex --galera-info`* is specified and *Percona XtraDB Cluster* is using `GTID` replication (version 5.6 only), **innobackupex** will execute `FLUSH BINARY LOGS` and then carry the current binary log as indicated in `SHOW MASTER STATUS` into the backup set. Bug fixed #1250375.

Percona XtraBackup did not roll back prepared XA transactions when applying the log. Which was a regression introduced with the original port of *Percona XtraBackup* patches to 5.6. Fixed by restoring code that has been lost in the port. Bug fixed #1254227.

Percona XtraBackup now uses `libgcrypt` built in randomization functions for setting the Initialization Vector. Bug fixed #1255300.

`xtrabackup_56` didn't support `ALL_O_DIRECT` option for innodb_flush_method in *Percona Server* 5.6. Bug fixed #1261877.

Other bugs fixed: #1255019, #1268325, #1265759, #1269694, #1271501.

Percona XtraBackup 2.1.8

Percona is glad to announce the release of *Percona XtraBackup* 2.1.8 on March 6th 2014. Downloads are available from our download site here and *Percona Software Repositories*.

This release is the current GA (Generally Available) stable release in the 2.1 series.

Bugs Fixed

Due to incorrect usage of `posix_fadvise()` hints, *Percona XtraBackup* discarded read-ahead buffers which resulted in higher `I/O` rate on the backup stage. Bug fixed #1093385.

Spurious trailing data blocks that would normally be ignored by *InnoDB* could lead to an assertion failure on the backup stage. Bug fixed #1177201.

A spurious warning message could cause issues with third-party wrapper scripts. Bug fixed #1271956.

xbcrypt could fail with the `xbcrypt:xb_crypt_read_chunk: unable to read chunk iv size at offset` error under some circumstances. Bug fixed #1273196.

xbstream could sometimes hang when extracting a broken or incomplete input stream. Bug fixed #1273207.

Preparing backups of *MySQL* or *Percona Server* 5.6 could raise an assertion failure in *Percona XtraBackup*. Bug fixed #1273468.

The downtime is decreased when the *`innobackupex --safe-slave-backup`* option is used to backup a replication slave. The SQL thread is now started before a temporary copy of redo log is streamed into the final location. Bug fixed #1284778.

Disabled the "binary version check" functionality in the VersionCheck module due to security concerns. Bug fixed #1285166.

Other bugs fixed: #1284078.

Percona XtraBackup 2.1.9

Percona is glad to announce the release of *Percona XtraBackup* 2.1.9 on May 7th 2014. Downloads are available from our download site here and *Percona Software Repositories*.

This release is the current GA (Generally Available) stable release in the 2.1 series.

New Features

Percona XtraBackup has now been rebased on *MySQL* 5.6.17.

Percona XtraBackup package is now available for *Ubuntu* 14.04.

Bugs Fixed

Percona XtraBackup couldn't be built with *Bison* 3.0. Bug fixed #1262439.

The xtrabackup binaries now recognize a new `my.cnf` option, *`open_files_limit`*. The effect is the same as for the server: it changes the maximum number of file descriptors available to the xtrabackup process. The actual limit depends on the platform and `ulimit` settings. Bug fixed #1183793.

If a remote *InnoDB* tablespace got CREATEd or ALTERed during the backup, an attempt to prepare such a backup later would lead to *Percona XtraBackup* crash. Bug fixed #1291299.

Code in both **innobackupex** and **xtrabackup**, that was supposed to make sure no child processes are left running in case **innobackupex** got killed or failed with an error, relied on the fact the `SIGTERM` and `SIGINT` signals were not blocked by the xtrabackup process. However, both `SIGTERM` and `SIGINT`

might be blocked by the process that had invoked **innobackupex**, for example, by the *Percona XtraDB Cluster* server processes doing an SST, in which case they were also blocked by the xtrabackup process, since the signal mask is inherited by child processes. Fixed by replacing `SIGTERM` in **innobackupex** and `SIGINT` in **xtrabackup** auto-termination with `SIGKILL`. Bug fixed #1294782.

A new table flag in the *InnoDB* data dictionary, introduced in *MySQL* 5.6.16, wasn't recognized by *Percona XtraBackup* which lead to `InnoDB: in InnoDB data dictionary has unknown flags 50.` warnings. Fixed by rebasing *Percona XtraBackup* on *MySQL* 5.6.17. Bug fixed #1302882.

All of Percona's software is open-source and free, all the details of the release can be found in the *2.1.9 milestone <https://launchpad.net/percona-xtrabackup/+milestone/2.1.9>* at Launchpad.

5.2.5 Percona *Percona XtraBackup* 2.2

Percona XtraBackup 2.2.1-alpha1

Percona is glad to announce the release of *Percona XtraBackup* 2.2.1-alpha1 on March 28th 2014. Downloads are available from our download site here. This **ALPHA** release, will be available in *Debian experimental* and *CentOS testing* repositories.

This is an **ALPHA** quality release and it is not intended for production. If you want a high quality, Generally Available release, the current Stable version should be used (currently 2.1.8 in the 2.1 series at the time of writing).

This release contains all of the features and bug fixes in *Percona XtraBackup 2.1.8*, plus the following:

New features

Percona XtraBackup has removed the multiple binaries (`xtrabackup_56`, `xtrabackup_55`, `xtrabackup`) and now uses single `xtrabackup` binary instead for handling backups. Single binary implementation removed the requirement to download server source tarballs and removed different patches which resulted in cleaner code.

Percona XtraBackup source layout has been changed to implement the single binary. *Percona XtraBackup* code can now be found in `storage/innobase/xtrabackup`.

Percona XtraBackup implemented support for Backup Locks.

Percona XtraBackup can now store *backup history* on the server itself in a special table created for that purpose.

`innobackupex-1.5.1` symlink has been removed, instead **innobackupex** binary should be used.

Percona XtraBackup has removed the `build.sh` script and it's now *built with CMake*.

Percona XtraBackup has been rebased on *MySQL* 5.6.16.

Bugs Fixed

Information about tool version used to take the backup was added by implementing *backup history* feature. Bug fixed #1133017.

If an XtraDB-based binary was used to a backup an *InnoDB* database, it will convert it to *XtraDB* by adding the `XTRADB_1` marker in the dictionary header page and by adding the `SYS_STATS` table. Bug fixed #988310.

Other bugs fixed: #721690, #1255899, #1255901, #1268300, and #788316.

Percona XtraBackup 2.2.2-beta1

Percona is glad to announce the release of *Percona XtraBackup* 2.2.2-beta1 on May 8th 2014. Downloads are available from our download site here. This **BETA** release will be available in *Debian experimental* and *CentOS testing* repositories.

This is a **BETA** quality release and it is not intended for production. If you want a high quality, Generally Available release, the current Stable version should be used (currently 2.1.9 in the 2.1 series at the time of writing).

New Features

Percona XtraBackup has now been rebased on *MySQL* 5.6.17.

Percona XtraBackup package is now available for *Ubuntu* 14.04.

Bugs Fixed

Percona XtraBackup couldn't be built with *Bison* 3.0. Bug fixed #1262439.

The xtrabackup binaries now recognize a new `my.cnf` option, *`open_files_limit`*. The effect is the same as for the server: it changes the maximum number of file descriptors available to the xtrabackup process. The actual limit depends on the platform and `ulimit` settings. Bug fixed #1183793.

If a remote *InnoDB* tablespace got CREATEd or ALTERed during the backup, an attempt to prepare such a backup later would lead to *Percona XtraBackup* crash. Bug fixed #1291299.

Code in both **innobackupex** and **xtrabackup**, that was supposed to make sure no child processes are left running in case **innobackupex** got killed or failed with an error, relied on the fact the `SIGTERM` and `SIGINT` signals were not blocked by the xtrabackup process. However, both `SIGTERM` and `SIGINT` might be blocked by the process that had invoked **innobackupex**, for example, by the *Percona XtraDB Cluster* server processes doing an SST, in which case they were also blocked by the xtrabackup process, since the signal mask is inherited by child processes. Fixed by replacing `SIGTERM` in **innobackupex** and `SIGINT` in **xtrabackup** auto-termination with `SIGKILL`. Bug fixed #1294782.

All of Percona's software is open-source and free, all the details of the release can be found in the 2.2.2-beta1 milestone at Launchpad.

Percona XtraBackup 2.2.3

Percona is glad to announce the release of *Percona XtraBackup* 2.2.3 on June 12th 2014. Downloads are available from our download site here and from *apt* and *yum* repositories.

This release is the first GA (Generally Available) stable release in the 2.2 series.

Bugs Fixed

Fixed the *InnoDB* redo log incompatibility with 5.1/5.5 server and compressed tables which was introduced by the upstream fix in *MySQL* 5.6.11 that could make *InnoDB* crash on recovery when replaying redo logs created on earlier versions. Bug fixed #1255476.

Percona XtraBackup did not flush the *InnoDB* REDO log buffer before finalizing the log copy. This would only become a problem when the binary log coordinates were used after restoring from a backup: the actual data files state after recovery could be inconsistent with the binary log coordinates. Bug fixed #1320685.

innobackupex now sets `wsrep_causal_reads` to `0` before executing `FLUSH TABLES WITH READ LOCK` if the server is a member of the Galera cluster. Bug fixed #1320441.

`storage/innobase/xtrabackup/CMakeLists.txt` now honors the `XB_DISTRIBUTION` environment variable when configuring `innobackupex.pl` to **innobackupex**. Bug fixed #1320856.

When backup locks are used, `xtrabackup_slave_info` should be written under `BINLOG` lock instead of `TABLE` lock. Bug fixed #1328532.

Other bugs fixed: #1318540.

All of Percona's software is open-source and free, all the details of the release can be found in the 2.2.3 milestone at Launchpad.

5.3 Glossary

LSN Each InnoDB page (usually 16kb in size) contains a log sequence number, or LSN. The LSN is the system version number for the entire database. Each page's LSN shows how recently it was changed.

innodb_file_per_table By default, all InnoDB tables and indexes are stored in the system tablespace on one file. This option causes the server to create one tablespace file per table. To enable it, set it on your configuration file,

```
[mysqld]
innodb_file_per_table
```

or start the server with `--innodb_file_per_table`.

innodb_expand_import This feature of *Percona Server* implements the ability to import arbitrary *.ibd* files exported using the *Percona XtraBackup* `--export` option.

See the the full documentation for more information.

innodb_data_home_dir The directory (relative to :term:‘ datadir‘) where the database server stores the files in a shared tablespace setup. This option does not affect the location of *innodb_file_per_table*. For example,

```
[mysqld]
innodb_data_home_dir = ./
```

innodb_data_file_path Specifies the names, sizes and location of shared tablespace files:

```
[mysqld]
innodb_data_file_path=ibdata1:50M;ibdata2:50M:autoextend
```

innodb_log_group_home_dir Specifies the location of the *InnoDB* log files:

```
[mysqld]
innodb_log_group_home=/var/lib/mysql
```

innodb_buffer_pool_size The size in bytes of the memory buffer to cache data and indexes of *InnoDB*‘s tables. This aims to reduce disk access to provide better performance. By default:

```
[mysqld]
innodb_buffer_pool_size=8MB
```

InnoDB Storage engine which provides ACID-compliant transactions and foreign key support, among others improvements over *MyISAM*. It is the default engine for *MySQL* as of the 5.5 series.

MyISAM Previous default storage engine for *MySQL* for versions prior to 5.5. It doesn't fully support transactions but in some scenarios may be faster than *InnoDB*. Each table is stored on disk in 3 files: *.frm*, *.MYD*, *.MYI*

XtraDB *Percona XtraDB* is an enhanced version of the InnoDB storage engine, designed to better scale on modern hardware, and including a variety of other features useful in high performance environments. It is fully backwards compatible, and so can be used as a drop-in replacement for standard InnoDB. More information here .

my.cnf This file refers to the database server's main configuration file. Most linux distributions place it as `/etc/mysql/my.cnf`, but the location and name depends on the particular installation. Note that this is not the only way of configuring the server, some systems does not have one even and rely on the command options to start the server and its defaults values.

datadir The directory in which the database server stores its databases. Most Linux distribution use `/var/lib/mysql` by default.

xbstream To support simultaneous compression and streaming, a new custom streaming format called xbstream was introduced to *Percona XtraBackup* in addition to the TAR format.

ibdata Default prefix for tablespace files, e.g. `ibdata1` is a 10MB autoextensible file that *MySQL* creates for the shared tablespace by default.

.frm For each table, the server will create a file with the `.frm` extension containing the table definition (for all storage engines).

.ibd On a multiple tablespace setup (*innodb_file_per_table* enabled), *MySQL* will store each newly created table on a file with a `.ibd` extension.

.MYD Each *MyISAM* table has `.MYD` (MYData) file which contains the data on it.

.MYI Each *MyISAM* table has `.MYI` (MYIndex) file which contains the table's indexes.

.exp Files with the `.exp` extension are created by *Percona XtraBackup* per each *InnoDB* tablespace when the *`xtrabckup --export`* option is used on prepare. These files can be used to import those tablespaces on *Percona Server* 5.5 or lower versions, see *restoring individual tables*"

.MRG Each table using the **MERGE** storage engine, besides of a *.frm* file, will have *.MRG* file containing the names of the *MyISAM* tables associated with it.

.TRG File containing the Triggers associated to a table, e.g. *:file:'mytable.TRG*. With the *.TRN* file, they represent all the Trigger definitions.

.TRN File containing the Triggers' Names associated to a table, e.g. *:file:'mytable.TRN*. With the *.TRG* file, they represent all the Trigger definitions.

.ARM Each table with the **Archive Storage Engine** has `.ARM` file which contains the metadata of it.

.ARZ Each table with the **Archive Storage Engine** has `.ARZ` file which contains the data of it.

.CSM Each table with the **CSV Storage Engine** has `.CSM` file which contains the metadata of it.

.CSV Each table with the **CSV Storage** engine has `.CSV` file which contains the data of it (which is a standard Comma Separated Value file).

.opt *MySQL* stores options of a database (like charset) in a file with a *`.opt`* extension in the database directory.

.par Each partitioned table has .par file which contains metadata about the partitions.

5.4 Index of files created by Percona XtraBackup

- Information related to the backup and the server
  - **`backup-my.cnf`** This file contains information to start the mini instance of InnoDB during the *`--apply-log`*. This is **NOT** a backup of original `my.cnf`. The InnoDB configuration is read from the file `backup-my.cnf` created by **innobackupex** when

the backup was made. *innobackupex --apply-log* uses InnoDB configuration from `backup-my.cnf` by default, or from *innobackupex --defaults-file*, if specified. InnoDB configuration in this context means server variables that affect data format, i.e. *innodb_page_size*, *innodb_log_block_size*, etc. Location-related variables, like *innodb_log_group_home_dir* or *innodb_data_file_path* are always ignored by *innobackupex --apply-log*, so preparing a backup always works with data files from the backup directory, rather than any external ones.

- **xtrabackup_checkpoints** The type of the backup (e.g. full or incremental), its state (e.g. prepared) and the *LSN* range contained in it. This information is used for incremental backups. Example of the `xtrabackup_checkpoints` after taking a full backup:

```
backup_type = full-backuped
from_lsn = 0
to_lsn = 15188961605
last_lsn = 15188961605
```

  Example of the `xtrabackup_checkpoints` after taking an incremental backup:

```
backup_type = incremental
from_lsn = 15188961605
to_lsn = 15189350111
last_lsn = 15189350111
```

- **xtrabackup_binlog_info** The binary log file used by the server and its position at the moment of the backup. Result of the **SHOW MASTER STATUS**.
- **xtrabackup_binlog_pos_innodb** The binary log file and its current position for *InnoDB* or *XtraDB* tables.
- **xtrabackup_binary** The **xtrabackup** binary used in the process.
- **xtrabackup_logfile** Contains data needed for running the: *--apply-log*. The bigger this file is the *--apply-log* process will take longer to finish.
- **<table_name>.delta.meta** This file is going to be created when performing the incremental backup. It contains the per-table delta metadata: page size, size of compressed page (if the value is 0 it means the tablespace isn't compressed) and space id. Example of this file could looks like this:

```
page_size = 16384
zip_size = 0
space_id = 0
```

- **<table_name>.ibd.pmap** This file contains ranges of skipped secondary index pages. These files are created only when doing the compact backup. The file format is a series of 2-value tuples, with each value being a 4-byte page offset corresponding to the first and the last endpoints of skipped ranges, respectively.

- Information related to the replication environment (if using the *--slave-info* option):
  - **xtrabackup_slave_info** The `CHANGE MASTER` statement needed for setting up a slave.
- Information related to the *Galera* and *Percona XtraDB Cluster* (if using the *--galera-info* option):
  - **xtrabackup_galera_info** Contains the values of status variables *wsrep_local_state_uuid* and *wsrep_last_committed*.

5.5 Trademark Policy

This Trademark Policy is to ensure that users of Percona-branded products or services know that what they receive has really been developed, approved, tested and maintained by Percona. Trademarks help to prevent confusion in the marketplace, by distinguishing one company's or person's products and services from another's.

Percona owns a number of marks, including but not limited to Percona, XtraDB, Percona XtraDB, XtraBackup, Percona XtraBackup, Percona Server, and Percona Live, plus the distinctive visual icons and logos associated with these marks. Both the unregistered and registered marks of Percona are protected.

Use of any Percona trademark in the name, URL, or other identifying characteristic of any product, service, website, or other use is not permitted without Percona's written permission with the following three limited exceptions.

First, you may use the appropriate Percona mark when making a nominative fair use reference to a bona fide Percona product.

Second, when Percona has released a product under a version of the GNU General Public License ("GPL"), you may use the appropriate Percona mark when distributing a verbatim copy of that product in accordance with the terms and conditions of the GPL.

Third, you may use the appropriate Percona mark to refer to a distribution of GPL-released Percona software that has been modified with minor changes for the sole purpose of allowing the software to operate on an operating system or hardware platform for which Percona has not yet released the software, provided that those third party changes do not affect the behavior, functionality, features, design or performance of the software. Users who acquire this Percona-branded software receive substantially exact implementations of the Percona software.

Percona reserves the right to revoke this authorization at any time in its sole discretion. For example, if Percona believes that your modification is beyond the scope of the limited license granted in this Policy or that your use of the Percona mark is detrimental to Percona, Percona will revoke this authorization. Upon revocation, you must immediately cease using the applicable Percona mark. If you do not immediately cease using the Percona mark upon revocation, Percona may take action to protect its rights and interests in the Percona mark. Percona does not grant any license to use any Percona mark for any other modified versions of Percona software; such use will require our prior written permission.

Neither trademark law nor any of the exceptions set forth in this Trademark Policy permit you to truncate, modify or otherwise use any Percona mark as part of your own brand. For example, if XYZ creates a modified version of the Percona Server, XYZ may not brand that modification as "XYZ Percona Server" or "Percona XYZ Server", even if that modification otherwise complies with the third exception noted above.

In all cases, you must comply with applicable law, the underlying license, and this Trademark Policy, as amended from time to time. For instance, any mention of Percona trademarks should include the full trademarked name, with proper spelling and capitalization, along with attribution of ownership to Percona LLC and/or its affiliates. For example, the full proper name for XtraBackup is Percona XtraBackup. However, it is acceptable to omit the word "Percona" for brevity on the second and subsequent uses, where such omission does not cause confusion.

In the event of doubt as to any of the conditions or exceptions outlined in this Trademark Policy, please contact trademarks@percona.com for assistance and we will do our very best to be helpful.

CHAPTER

SIX

INDICES AND TABLES

- *genindex*
- *search*

Symbols

–apply-log
innobackupex command line option, 34
–apply-log-only
xtrabackup command line option, 57
–backup
xtrabackup command line option, 57
–compact
innobackupex command line option, 34
xtrabackup command line option, 58
–compress
innobackupex command line option, 34
xtrabackup command line option, 58
–compress-chunk-size=#
innobackupex command line option, 34
xtrabackup command line option, 58
–compress-threads=#
innobackupex command line option, 34
xtrabackup command line option, 58
–copy-back
innobackupex command line option, 34
–create-ib-logfile
xtrabackup command line option, 58
–databases=LIST
innobackupex command line option, 34
–datadir=DIRECTORY
xtrabackup command line option, 58
–decompress
innobackupex command line option, 34
–decrypt=ENCRYPTION-ALGORITHM
innobackupex command line option, 34
–defaults-extra-file=[MY.CNF]
innobackupex command line option, 35
xtrabackup command line option, 58
–defaults-file=[MY.CNF]
innobackupex command line option, 34
xtrabackup command line option, 58
–defaults-group=GROUP-NAME
innobackupex command line option, 35
xtrabackup command line option, 58
–encrypt-chunk-size=#
innobackupex command line option, 35
–encrypt-key-file=ENCRYPTION_KEY_FILE
innobackupex command line option, 35
–encrypt-key=ENCRYPTION_KEY
innobackupex command line option, 35
–encrypt-threads=#
innobackupex command line option, 35
–encrypt=ENCRYPTION_ALGORITHM
innobackupex command line option, 35
–export
innobackupex command line option, 35
xtrabackup command line option, 58
–extra-lsndir=DIRECTORY
innobackupex command line option, 35
xtrabackup command line option, 58
–force-non-empty-directories
innobackupex command line option, 35
–galera-info
innobackupex command line option, 35
–help
innobackupex command line option, 35
–history=NAME
innobackupex command line option, 35
–host=HOST
innobackupex command line option, 36
–ibbackup=IBBACKUP-BINARY
innobackupex command line option, 36
–include=REGEXP
innobackupex command line option, 36
–incremental
innobackupex command line option, 36
–incremental-basedir=DIRECTORY
innobackupex command line option, 36
xtrabackup command line option, 58
–incremental-dir=DIRECTORY
innobackupex command line option, 36
xtrabackup command line option, 58
–incremental-force-scan
xtrabackup command line option, 58
–incremental-history-name=NAME
innobackupex command line option, 36
–incremental-history-uuid=UUID
innobackupex command line option, 36

–incremental-lsn=LSN
innobackupex command line option, 36
xtrabackup command line option, 58
–innodb-log-arch-dir=DIRECTORY
xtrabackup command line option, 59
–innodb-miscellaneous
xtrabackup command line option, 59
–kill-long-queries-timeout=SECONDS
innobackupex command line option, 37
–kill-long-query-type=all|select
innobackupex command line option, 37
–lock-wait-query-type=all|update
innobackupex command line option, 37
–lock-wait-threshold=SECONDS
innobackupex command line option, 37
–lock-wait-timeout=SECONDS
innobackupex command line option, 37
–log-copy-interval=#
innobackupex command line option, 37
xtrabackup command line option, 59
–log-stream
xtrabackup command line option, 59
–move-back
innobackupex command line option, 37
–no-defaults
xtrabackup command line option, 59
–no-lock
innobackupex command line option, 37
–no-timestamp
innobackupex command line option, 37
–no-version-check
innobackupex command line option, 37
–parallel=NUMBER-OF-THREADS
innobackupex command line option, 37
–parallel=#
xtrabackup command line option, 59
–password=PASSWORD
innobackupex command line option, 38
–port=PORT
innobackupex command line option, 38
–prepare
xtrabackup command line option, 59
–print-defaults
xtrabackup command line option, 60
–print-param
xtrabackup command line option, 60
–rebuild-indexes
innobackupex command line option, 38
–rebuild-threads=NUMBER-OF-THREADS
innobackupex command line option, 38
–rebuild_indexes
xtrabackup command line option, 60
–rebuild_threads=#
xtrabackup command line option, 60
–redo-only
innobackupex command line option, 38
–rsync
innobackupex command line option, 38
–safe-slave-backup
innobackupex command line option, 38
–safe-slave-backup-timeout=SECONDS
innobackupex command line option, 38
–scpopt = SCP-OPTIONS
innobackupex command line option, 38
–slave-info
innobackupex command line option, 38
–socket
innobackupex command line option, 38
–sshopt=SSH-OPTIONS
innobackupex command line option, 38
–stats
xtrabackup command line option, 60
–stream=STREAMNAME
innobackupex command line option, 39
–stream=name
xtrabackup command line option, 60
–suspend-at-end
xtrabackup command line option, 60
–tables-file=FILE
innobackupex command line option, 39
–tables-file=name
xtrabackup command line option, 60
–tables=name
xtrabackup command line option, 60
–target-dir=DIRECTORY
xtrabackup command line option, 60
–throttle=IOS
innobackupex command line option, 39
–throttle=#
xtrabackup command line option, 60
–tmpdir=DIRECTORY
innobackupex command line option, 39
–tmpdir=name
xtrabackup command line option, 60
–to-archived-lsn=LSN
xtrabackup command line option, 60
–use-memory=#
innobackupex command line option, 39
xtrabackup command line option, 60
–user=USER
innobackupex command line option, 39
–version
innobackupex command line option, 39
xtrabackup command line option, 61
–version-check
innobackupex command line option, 39
-a, –encrypt-algo=name
command line option, 61

-d, –decrypt
    command line option, 61
-f, –encrypt-key-file=name
    command line option, 61
-i, –input=name
    command line option, 61
-k, –encrypt-key=name
    command line option, 61
-o, –output=name
    command line option, 61
-s, –encrypt-chunk-size=#
    command line option, 61
-v, –verbose
    command line option, 62
.ARM, **116**
.ARZ, **116**
.CSM, **116**
.CSV, **116**
.MRG, **116**
.MYD, **116**
.MYI, **116**
.TRG, **116**
.TRN, **116**
.exp, **116**
.frm, **116**
.ibd, **116**
.opt, **116**
.par, **116**

C

command line option
    -a, –encrypt-algo=name, 61
    -d, –decrypt, 61
    -f, –encrypt-key-file=name, 61
    -i, –input=name, 61
    -k, –encrypt-key=name, 61
    -o, –output=name, 61
    -s, –encrypt-chunk-size=#, 61
    -v, –verbose, 62

D

datadir, **116**

I

ibdata, **116**
innobackupex command line option
    –apply-log, 34
    –compact, 34
    –compress, 34
    –compress-chunk-size=#, 34
    –compress-threads=#, 34
    –copy-back, 34
    –databases=LIST, 34
    –decompress, 34
    –decrypt=ENCRYPTION-ALGORITHM, 34
    –defaults-extra-file=[MY.CNF], 35
    –defaults-file=[MY.CNF], 34
    –defaults-group=GROUP-NAME, 35
    –encrypt-chunk-size=#, 35
    –encrypt-key-file=ENCRYPTION_KEY_FILE, 35
    –encrypt-key=ENCRYPTION_KEY, 35
    –encrypt-threads=#, 35
    –encrypt=ENCRYPTION_ALGORITHM, 35
    –export, 35
    –extra-lsndir=DIRECTORY, 35
    –force-non-empty-directories, 35
    –galera-info, 35
    –help, 35
    –history=NAME, 35
    –host=HOST, 36
    –ibbackup=IBBACKUP-BINARY, 36
    –include=REGEXP, 36
    –incremental, 36
    –incremental-basedir=DIRECTORY, 36
    –incremental-dir=DIRECTORY, 36
    –incremental-history-name=NAME, 36
    –incremental-history-uuid=UUID, 36
    –incremental-lsn=LSN, 36
    –kill-long-queries-timeout=SECONDS, 37
    –kill-long-query-type=all|select, 37
    –lock-wait-query-type=all|update, 37
    –lock-wait-threshold=SECONDS, 37
    –lock-wait-timeout=SECONDS, 37
    –log-copy-interval=#, 37
    –move-back, 37
    –no-lock, 37
    –no-timestamp, 37
    –no-version-check, 37
    –parallel=NUMBER-OF-THREADS, 37
    –password=PASSWORD, 38
    –port=PORT, 38
    –rebuild-indexes, 38
    –rebuild-threads=NUMBER-OF-THREADS, 38
    –redo-only, 38
    –rsync, 38
    –safe-slave-backup, 38
    –safe-slave-backup-timeout=SECONDS, 38
    –scpopt = SCP-OPTIONS, 38
    –slave-info, 38
    –socket, 38
    –sshopt=SSH-OPTIONS, 38
    –stream=STREAMNAME, 39
    –tables-file=FILE, 39
    –throttle=IOS, 39
    –tmpdir=DIRECTORY, 39
    –use-memory=#, 39
    –user=USER, 39
    –version, 39

–version-check, 39
InnoDB, **115**
innodb_buffer_pool_size, **115**
innodb_data_file_path, **115**
innodb_data_home_dir, **115**
innodb_expand_import, **115**
innodb_file_per_table, **115**
innodb_log_group_home_dir, **115**

L

LSN, **115**

M

my.cnf, **116**
MyISAM, **115**

X

xbstream, **116**
xtrabackup command line option
–apply-log-only, 57
–backup, 57
–compact, 58
–compress, 58
–compress-chunk-size=#, 58
–compress-threads=#, 58
–create-ib-logfile, 58
–datadir=DIRECTORY, 58
–defaults-extra-file=[MY.CNF], 58
–defaults-file=[MY.CNF], 58
–defaults-group=GROUP-NAME, 58
–export, 58
–extra-lsndir=DIRECTORY, 58
–incremental-basedir=DIRECTORY, 58
–incremental-dir=DIRECTORY, 58
–incremental-force-scan, 58
–incremental-lsn=LSN, 58
–innodb-log-arch-dir=DIRECTORY, 59
–innodb-miscellaneous, 59
–log-copy-interval=#, 59
–log-stream, 59
–no-defaults, 59
–parallel=#, 59
–prepare, 59
–print-defaults, 60
–print-param, 60
–rebuild_indexes, 60
–rebuild_threads=#, 60
–stats, 60
–stream=name, 60
–suspend-at-end, 60
–tables-file=name, 60
–tables=name, 60
–target-dir=DIRECTORY, 60
–throttle=#, 60
–tmpdir=name, 60
–to-archived-lsn=LSN, 60
–use-memory=#, 60
–version, 61
XtraDB, **116**